In [2]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('Suicides with reason.csv')
data
```

Out[2]:

	State	Year	Type_code	Type	Gender	Age_group	Total
0	A & N ISLANDS	2001	Causes	Cancer	Male	15-29	0
1	A & N ISLANDS	2001	Causes	Divorce	Male	60+	0
2	A & N ISLANDS	2001	Causes	Dowry Dispute	Female	60+	0
3	A & N ISLANDS	2001	Causes	Ideological Causes/Hero Worshipping	Female	60+	0
4	A & N ISLANDS	2001	Causes	Illness (Aids/STD)	Female	0-14	0
...	...	...	...	...	...	...	...
237356	WEST BENGAL	2012	Professional_Profile	Professional Activity	Male	60+	0
237357	WEST BENGAL	2012	Professional_Profile	Self-employed (Business activity)	Male	0-14	0
237358	WEST BENGAL	2012	Professional_Profile	Service (Government)	Male	15-29	0
237359	WEST BENGAL	2012	Professional_Profile	Service (Government)	Male	60+	0
237360	WEST BENGAL	2012	Social_Status	Never Married	Male	0-100+	2658

237361 rows × 7 columns

In [3]:

```python
data.shape
```

Out[3]:

```
(237361, 7)
```

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237361 entries, 0 to 237360
Data columns (total 7 columns):
 #   Column     Non-Null Count   Dtype 
---  ------     --------------   ----- 
 0   State      237361 non-null  object
 1   Year       237361 non-null  int64 
 2   Type_code  237361 non-null  object
 3   Type       237361 non-null  object
 4   Gender     237361 non-null  object
 5   Age_group  237361 non-null  object
 6   Total      237361 non-null  int64 
dtypes: int64(2), object(5)
memory usage: 12.7+ MB
```

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
State        0
Year         0
Type_code    0
Type         0
Gender       0
Age_group    0
Total        0
dtype: int64
```

In [6]:
```
data['Total']
```

Out[6]:
```
0            0
1            0
2            0
3            0
4            0
          ... 
237356       0
237357       0
237358       0
237359       0
237360    2658
Name: Total, Length: 237361, dtype: int64
```

In [7]:
```
data.Total.sum()
```

Out[7]:
```
11962175
```

In [8]:
```
index = data['State']=='ASSAM'
data[index].Total.sum()
```

Out[8]:
```
172276
```

In [22]:
```
statewise = data.groupby(['State','Age_group','Gender','Year',]).Total.sum().d
rop(labels='TOTAL (ALL INDIA)').drop(labels='TOTAL (STATES)').drop(labels='TOT
AL (UTs)')
statewise
```

Out[22]:
```
State          Age_group  Gender  Year
A & N ISLANDS  0-100+     Female  2001     100
                                  2002     106
                                  2003      86
                                  2004      82
                                  2005     106
                                          ...
WEST BENGAL    60+        Male    2008    1104
                                  2009    1389
                                  2010    1674
                                  2011    2193
                                  2012     665
Name: Total, Length: 5040, dtype: int64
```

In [23]:
```
statewise.to_csv('statewise.csv')
```

In [24]:
```
filterdata = pd.read_csv('statewise.csv')
```

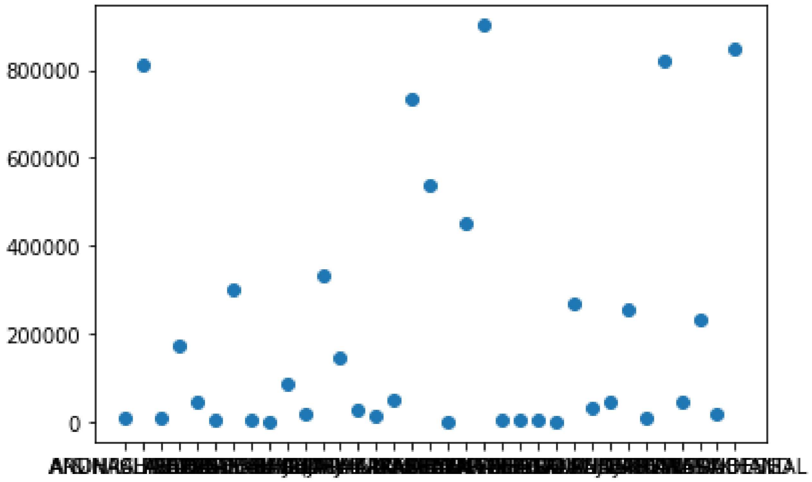
In [19]:
```
D = filterdata.groupby(['State']).Total.sum()
D
```

Out[19]:
```
State
A & N ISLANDS          8109
ANDHRA PRADESH       814059
ARUNACHAL PRADESH      6633
ASSAM                172276
BIHAR                 46214
CHANDIGARH             5164
CHHATTISGARH         302354
D & N HAVELI           3430
DAMAN & DIU            1391
DELHI (UT)            84272
GOA                   17363
GUJARAT              330858
HARYANA              147176
HIMACHAL PRADESH      26562
JAMMU & KASHMIR       14821
JHARKHAND             49720
KARNATAKA            734825
KERALA               538946
LAKSHADWEEP              50
MADHYA PRADESH       451535
MAHARASHTRA          901945
MANIPUR                2102
MEGHALAYA              5415
MIZORAM                4154
NAGALAND               1728
ODISHA               267234
PUDUCHERRY            32144
PUNJAB                46350
RAJASTHAN            255134
SIKKIM                 9606
TAMIL NADU           818691
TRIPURA               45965
UTTAR PRADESH        233352
UTTARAKHAND           18496
WEST BENGAL          849936
Name: Total, dtype: int64
```

In [20]:
```
x = D.index.values
y = D.values
```

In [21]:

```
plt.scatter(x,y)
plt.show()
```



In [27]:

```
YD = filterdata.groupby(['Year']).Total.sum()
YD
```

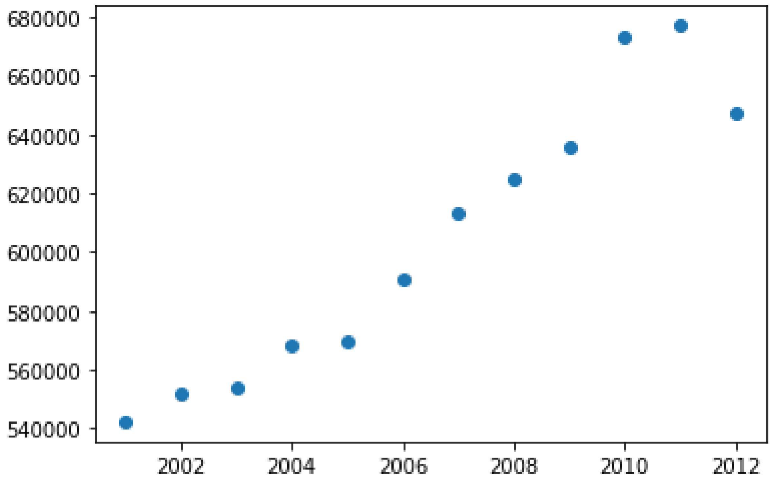
Out[27]:

```
Year
2001    542440
2002    551980
2003    554218
2004    568349
2005    569545
2006    590543
2007    613119
2008    625014
2009    635429
2010    672926
2011    677159
2012    647288
Name: Total, dtype: int64
```

In [32]:

```
x1 = YD.index.values
y1 = YD.values
```

In [33]:
```
plt.scatter(x1,y1)
plt.show()
```



In [34]:
```
filterdata.groupby('Gender').Total.sum()
```

Out[34]:
```
Gender
Female    2606922
Male      4641088
Name: Total, dtype: int64
```

In [35]:
```
data.groupby(['State','Year']).Total.sum().drop(labels='TOTAL (ALL INDIA)').drop(labels='TOTAL (STATES)').drop(labels='TOTAL (UTs)')
```

Out[35]:
```
State          Year
A & N ISLANDS  2001      645
               2002      720
               2003      565
               2004      610
               2005      695
                        ...
WEST BENGAL    2008    74260
               2009    73240
               2010    80185
               2011    82460
               2012    44871
Name: Total, Length: 420, dtype: int64
```

In [36]:
```
data.groupby(['State','Year'])['Total'].sum()
```

Out[36]:
```
State           Year
A & N ISLANDS   2001      645
                2002      720
                2003      565
                2004      610
                2005      695
                        ...
WEST BENGAL     2008    74260
                2009    73240
                2010    80185
                2011    82460
                2012    44871
Name: Total, Length: 456, dtype: int64
```

In [37]:
```
data.groupby(['State','Gender'])[['Total']].sum().drop(labels='TOTAL (ALL INDIA)').drop(labels='TOTAL (STATES)').drop(labels='TOTAL (UTs)')
```

Out[37]:

		Total
State	**Gender**	
A & N ISLANDS	**Female**	2750
	Male	5359
ANDHRA PRADESH	**Female**	271939
	Male	542120
ARUNACHAL PRADESH	**Female**	1954
...	**...**	...
UTTAR PRADESH	**Male**	125327
UTTARAKHAND	**Female**	7548
	Male	10948
WEST BENGAL	**Female**	365241
	Male	484695

70 rows × 1 columns

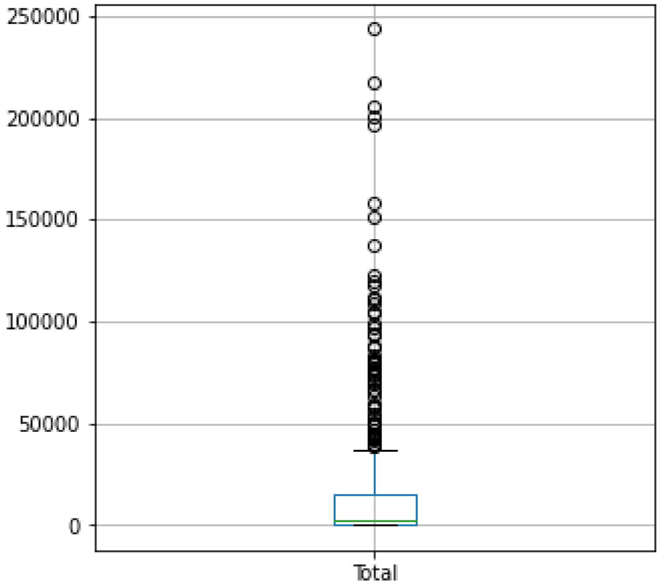
In [38]:
```
newdata = data.groupby(['State','Gender','Age_group'])[['Total']].sum().drop(labels='TOTAL (ALL INDIA)').drop(labels='TOTAL (STATES)').drop(labels='TOTAL (UTs)')
newdata
```

Out[38]:

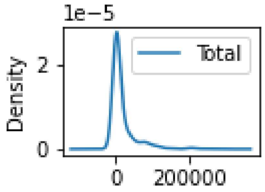
			Total
State	**Gender**	**Age_group**	
A & N ISLANDS	**Female**	**0-100+**	1100
		0-14	93
		15-29	894
		30-44	399
		45-59	171
...	**...**	**...**	...
WEST BENGAL	**Male**	**0-14**	7895
		15-29	86688
		30-44	103877
		45-59	66370
		60+	19043

420 rows × 1 columns

In [39]:
```
newdata.boxplot(figsize=(5,5))
plt.show()
```



In [42]:
```
newdata.plot(kind='density', subplots=True, layout=(3,3), sharex=False)
plt.show()
```



In [43]:
```
newdata.median()
```

Out[43]:
```
Total    2270.5
dtype: float64
```

In [44]:
```
newdata.to_csv('decision.csv')
```

In [45]:
```
data.groupby(['Year','Gender'])[['Total']].sum()
```

Out[45]:

		Total
Year	**Gender**	
2001	**Female**	356253
	Male	541653
2002	**Female**	341512
	Male	617774
2003	**Female**	355586
	Male	521318
2004	**Female**	346672
	Male	638465
2005	**Female**	342231
	Male	582379
2006	**Female**	370826
	Male	571163
2007	**Female**	271682
	Male	652382
2008	**Female**	395534
	Male	706107
2009	**Female**	367126
	Male	646641
2010	**Female**	400571
	Male	726795
2011	**Female**	383042
	Male	743522
2012	**Female**	359515
	Male	723426

In [47]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

deriveddata = pd.read_csv('num_decision3.csv')
deriveddata
```

Out[47]:

	State	Gender	Age_group	Tendency
0	0	0	0	1
1	0	0	1	0
2	0	0	2	1
3	0	0	3	0
4	0	0	4	0
...	...	...	...	...
415	34	1	1	0
416	34	1	2	1
417	34	1	3	1
418	34	1	4	0
419	34	1	5	0

420 rows × 4 columns

In [48]:

```
# Separating the target variable
X = deriveddata.values[:, 0:3]
Y = deriveddata.values[:, 3]

# Splitting the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(
X, Y, test_size = 0.3, random_state = 100)
```

In [49]:

```
print(X)
```

```
[[ 0  0  0]
 [ 0  0  1]
 [ 0  0  2]
 ...
 [34  1  3]
 [34  1  4]
 [34  1  5]]
```

In [50]:
```
clf_gini = DecisionTreeClassifier(criterion = "gini",
random_state = 100,max_depth=3, min_samples_leaf=5)
clf_gini.fit(X_train, y_train)
```

Out[50]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=3, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=5, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=100, splitter='best')
```

In [51]:
```
clf_entropy = DecisionTreeClassifier(
criterion = "entropy", random_state = 100,max_depth = 3, min_samples_leaf = 5)
clf_entropy.fit(X_train, y_train)
```

Out[51]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=3, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=5, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=100, splitter='best')
```

In [52]:
```
# Prediction using gini
print("Results Using Gini Index:")
y_pred_gini = clf_gini.predict(X_test)
print("Predicted values:")
print(y_pred_gini)

print("Confusion Matrix: ",
confusion_matrix(y_test, y_pred_gini))

print ("Accuracy : ",
accuracy_score(y_test,y_pred_gini)*100)

print("Report : ",
classification_report(y_test, y_pred_gini))
```

```
Results Using Gini Index:
Predicted values:
[0 0 1 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0 1 0 0 1 0
 1 1 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 1
 0 1 0 0 0 0 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1
 0 1 1 0 0 0 0 1 1 0 1 1 0 0 0]
Confusion Matrix:  [[60  8]
 [ 4 54]]
Accuracy :  90.47619047619048
Report :                precision    recall  f1-score   support

           0       0.94      0.88      0.91        68
           1       0.87      0.93      0.90        58

    accuracy                           0.90       126
   macro avg       0.90      0.91      0.90       126
weighted avg       0.91      0.90      0.90       126
```

In [53]:
```
y_pred_gini = clf_gini.predict([[24, 1, 3]])
print("Predicted values:")
print(y_pred_gini)
```

```
Predicted values:
[1]
```

In [54]:
```
print("Results Using Entropy:")
y_pred_entropy = clf_entropy.predict(X_test)
print("Predicted values:")
print(y_pred_entropy)

print("Confusion Matrix: ",
confusion_matrix(y_test, y_pred_entropy))

print ("Accuracy : ",
accuracy_score(y_test,y_pred_entropy)*100)

print("Report : ",
classification_report(y_test, y_pred_entropy))
```

```
Results Using Entropy:
Predicted values:
[0 0 1 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0 1 0 0 1 0
 1 1 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1
 0 1 0 0 0 0 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1
 0 1 1 0 0 0 0 1 1 0 1 1 0 0 0]
Confusion Matrix:  [[60  8]
 [ 4 54]]
Accuracy :  90.47619047619048
Report :                precision    recall  f1-score   support

           0       0.94      0.88      0.91        68
           1       0.87      0.93      0.90        58

    accuracy                           0.90       126
   macro avg       0.90      0.91      0.90       126
weighted avg       0.91      0.90      0.90       126
```

In [55]:
```
y_pred_entropy = clf_entropy.predict([[20, 0, 1]])
print("Predicted values:")
print(y_pred_entropy)
```

```
Predicted values:
[0]
```

In [56]:
```
import joblib
joblib.dump(clf_entropy, 'Decision_Tree_Entropy.pkl', compress=9)
joblib.dump(clf_gini, 'Decision_Tree_Gini.pkl', compress=9)
```

Out[56]:
```
['Decision_Tree_Gini.pkl']
```

In [57]:
```
#import joblib
#model_clone = joblib.load('my_model.pkl
```