Property stolen and recoverd

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import os
for dirname, _, filenames in os.walk('/content/drive/MyDrive/Data visualozation/Crime'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

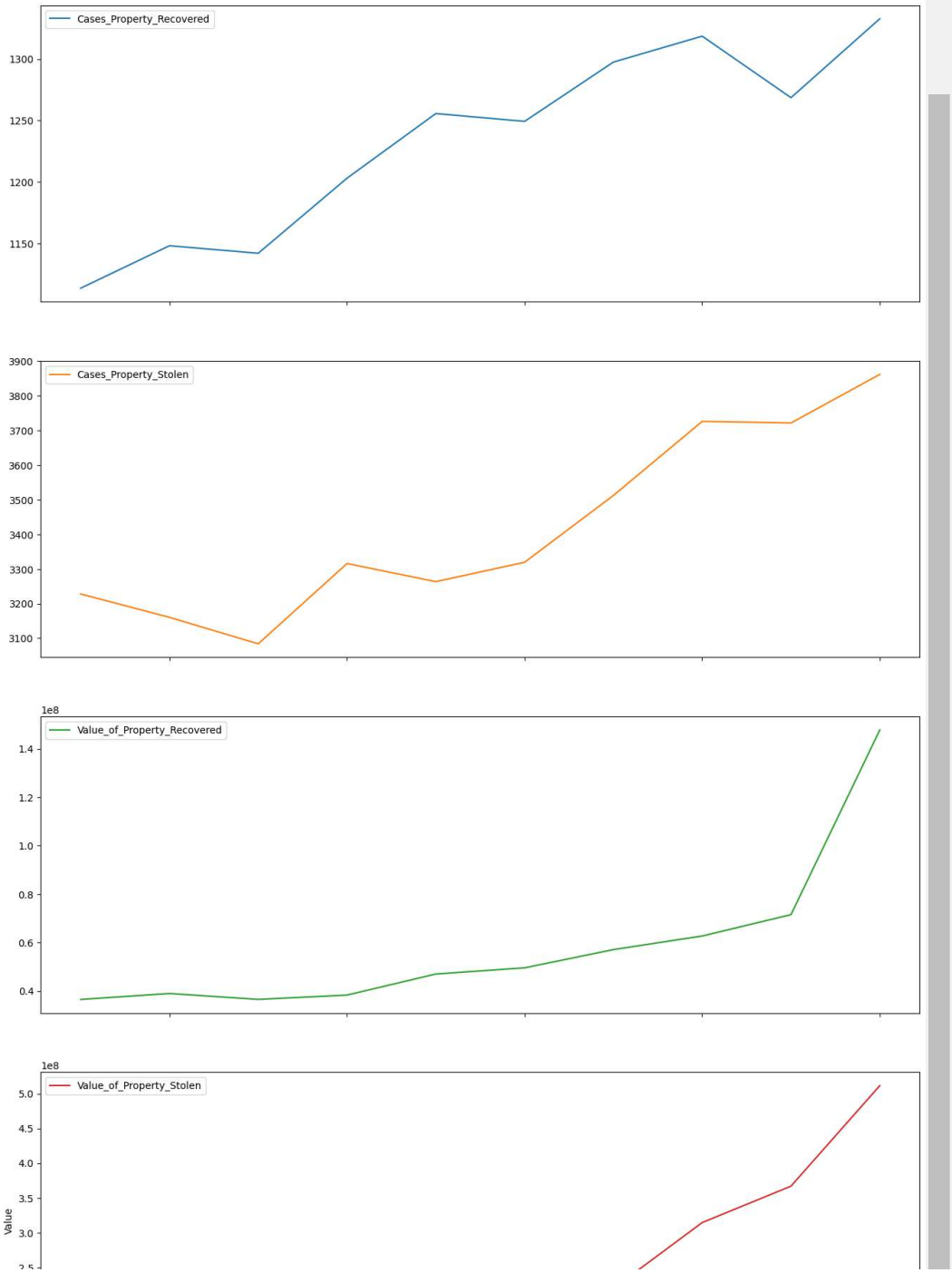
```
/content/drive/MyDrive/Data visualozation/Crime/Auto_theft.csv
/content/drive/MyDrive/Data visualozation/Crime/Complaints_against_police.csv
/content/drive/MyDrive/Data visualozation/Crime/Property_stolen_and_recovered.csv
/content/drive/MyDrive/Data visualozation/Crime/Rape_Victims.csv
/content/drive/MyDrive/Data visualozation/Crime/Murders.csv
/content/drive/MyDrive/Data visualozation/Crime/Indian map/India States/Indian_states.prj
/content/drive/MyDrive/Data visualozation/Crime/Indian map/India States/Indian_states.shp
/content/drive/MyDrive/Data visualozation/Crime/Indian map/India States/Indian_states.dbf
/content/drive/MyDrive/Data visualozation/Crime/Indian map/India States/Indian_states.shx
/content/drive/MyDrive/Data visualozation/Crime/Indian map/India Boundary/India_boundary.shx
/content/drive/MyDrive/Data visualozation/Crime/Indian map/India Boundary/India_boundary.prj
/content/drive/MyDrive/Data visualozation/Crime/Indian map/India Boundary/India_boundary.shp
/content/drive/MyDrive/Data visualozation/Crime/Indian map/India Boundary/India_boundary.dbf
/content/drive/MyDrive/Data visualozation/Crime/Murged_data/output1.csv
/content/drive/MyDrive/Data visualozation/Crime/Murged_data/output2.csv
```

```
property=pd.read_csv("/content/drive/MyDrive/Data visualozation/Crime/Property_stolen_and_recovered.csv")
property
```

	Area_Name	Year	Group_Name	Sub_Group_Name	Cases_Property_Recovered	Cases_Property_Stolen	Va
0	Andaman & Nicobar Islands	2001	Burglary - Property	3. Burglary	27	64	
1	Andhra Pradesh	2001	Burglary - Property	3. Burglary	3321	7134	
2	Arunachal Pradesh	2001	Burglary - Property	3. Burglary	66	248	
3	Assam	2001	Burglary - Property	3. Burglary	539	2423	
4	Bihar	2001	Burglary - Property	3. Burglary	367	3231	
...	...	...	...	...	...	...	
2444	Tamil Nadu	2010	Total Property	7. Total Property Stolen & Recovered	16125	21509	
2445	Tripura	2010	Total Property	7. Total Property Stolen & Recovered	192	879	
2446	Uttar Pradesh	2010	Total Property	7. Total Property Stolen & Recovered	9130	35068	
2447	Uttarakhand	2010	Total Property	7. Total Property Stolen & Recovered	964	2234	
2448	West Bengal	2010	Total Property	7. Total Property Stolen &	4548	23759	

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
# Group the data by year and calculate the mean of each numerical column
grouped = property.groupby('Year').mean()

# Plot each numerical column in the same plot
grouped.plot(kind='line', subplots=True, figsize=(16, 25))

# Set the plot title and labels
plt.suptitle('Numerical data by year')
plt.xlabel('Year')
plt.ylabel('Value')

# Show the plot
plt.show()
```



This is a balanced datasets with almost equal instances from different states

```
property.Area_Name.value_counts()
```

```
Andaman & Nicobar Islands    70
Puducherry                   70
Maharashtra                  70
Manipur                      70
Meghalaya                    70
Mizoram                      70
Nagaland                     70
Odisha                       70
Punjab                       70
Andhra Pradesh               70
Rajasthan                    70
Sikkim                       70
Tamil Nadu                   70
Tripura                      70
Uttar Pradesh                70
```

```
Uttarakhand                 70
Madhya Pradesh              70
Kerala                      70
Delhi                       70
Daman & Diu                 70
Arunachal Pradesh           70
Assam                       70
Bihar                       70
Chandigarh                  70
Chhattisgarh                70
Dadra & Nagar Haveli        70
West Bengal                 70
Karnataka                   70
Goa                         70
Gujarat                     70
Haryana                     70
Himachal Pradesh            70
Jammu & Kashmir             70
Jharkhand                   70
Lakshadweep                 69
Name: Area_Name, dtype: int64
```

Same goes for the Year column

```
property.Year.value_counts()
```

```
2001    245
2002    245
2003    245
2004    245
2005    245
2006    245
2008    245
2009    245
2010    245
2007    244
Name: Year, dtype: int64
```

```
property.Group_Name.value_counts()
```

```
Burglary - Property                   350
Criminal Breach of Trust - Property   350
Dacoity -Property                     350
Other heads of Property               350
Robbery - Property                    350
Theft - Property                      350
Total Property                        349
Name: Group_Name, dtype: int64
```

Group Name and Sub Group Name have the same number of instances per value type so they can be eliminated

```
property.Sub_Group_Name.value_counts()
```

```
3. Burglary                            350
5. Criminal Breach of Trust            350
1. Dacoity                             350
6. Other Property                      350
2. Robbery                             350
4. Theft                               350
7. Total Property Stolen & Recovered   349
Name: Sub_Group_Name, dtype: int64
```

This groupby function further concretize our notion about Group Name and Sub Group Name

```
a=property.groupby(['Group_Name']).get_group('Robbery - Property')
a.Sub_Group_Name.value_counts()
```

```
2. Robbery    350
Name: Sub_Group_Name, dtype: int64
```

Since all the values of Sub Group Name from the Group Name ='Robbery - Property' are '2. Robbery' we can safely delete Sub_Group_Name from the dataset

```
a=property.groupby(['Group_Name']).get_group('Total Property')
a.Sub_Group_Name.value_counts()
```

```
7. Total Property Stolen & Recovered    349
Name: Sub_Group_Name, dtype: int64
```

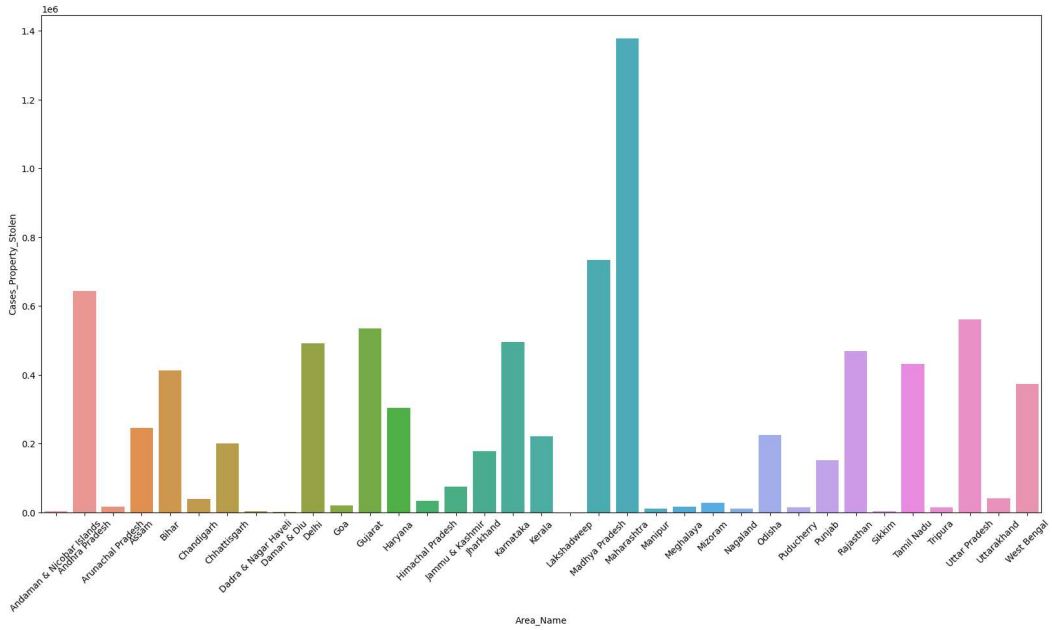
Checked the Group Name and Sub Group Name similarity for all the values of Group Name

```
#Since Sub_Group_Name is an irrelevant variable as previously proven ,we will use property1 as the base dataset
property1=property.drop(['Sub_Group_Name'],axis=1)
```

Lets group the dataset based on the Area_Name(States) and drop the Year column Adjusting the Plot Size Using Seaborn to plot the Bar Plot Graph for Area_Name and Cases_Property_Stolen Rotating the labels of the graph for better visibility

```
property_bystate=property1.groupby(['Area_Name'],as_index=False).sum()
property_bystate.drop("Year",axis=1,inplace=True)
plt.figure(figsize = (20, 10))
chart=sns.barplot(x=property_bystate.Area_Name,y=property_bystate.Cases_Property_Stolen)
chart.set_xticklabels(chart.get_xticklabels(), rotation=45)
chart
```

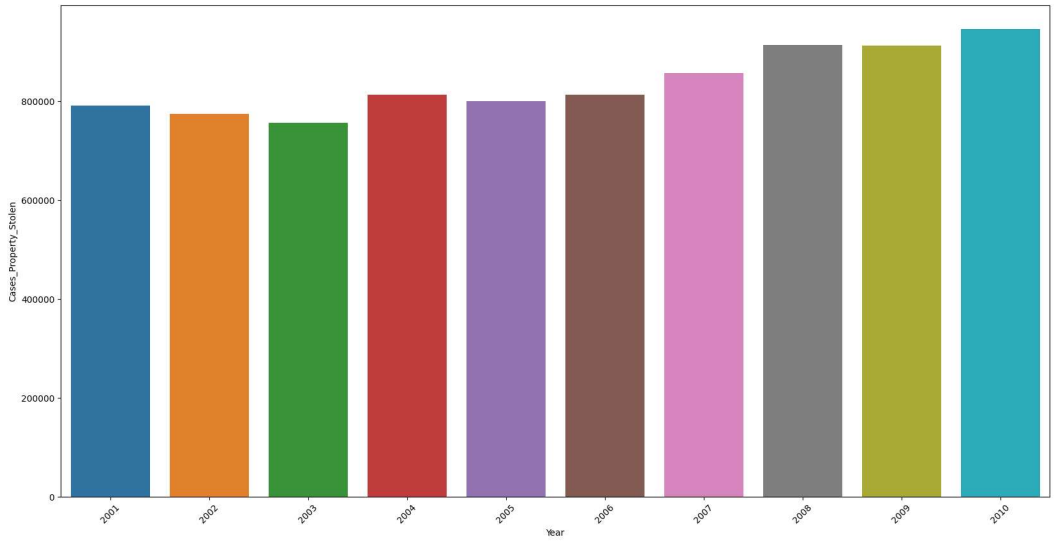
```
<ipython-input-12-dbb5afe8a853>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy
  property_bystate=property1.groupby(['Area_Name'],as_index=False).sum()
<Axes: xlabel='Area_Name', ylabel='Cases_Property_Stolen'>
```




```
#Cases of Property Stolen across the year of all the States
sortbyyear=property1.groupby(['Year'],as_index=False).sum()
```

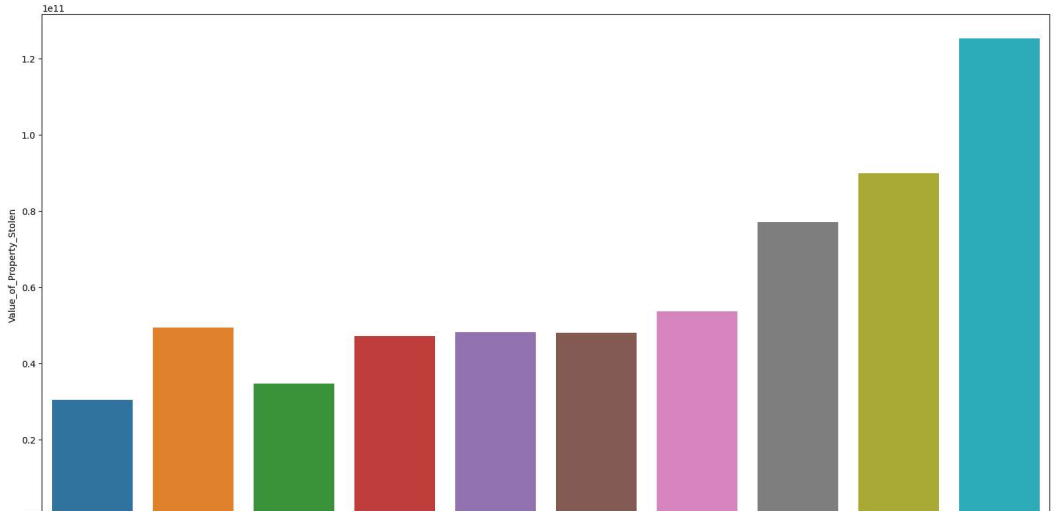
```
sortbyyear
plt.figure(figsize = (20, 10))
chart=sns.barplot(x=sortbyyear.Year,y=sortbyyear.Cases_Property_Stolen)
chart.set_xticklabels(chart.get_xticklabels(), rotation=45)
chart
```

```
<ipython-input-13-f655575becb2>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy
  sortbyyear=property1.groupby(['Year'],as_index=False).sum()
<Axes: xlabel='Year', ylabel='Cases_Property_Stolen'>
```



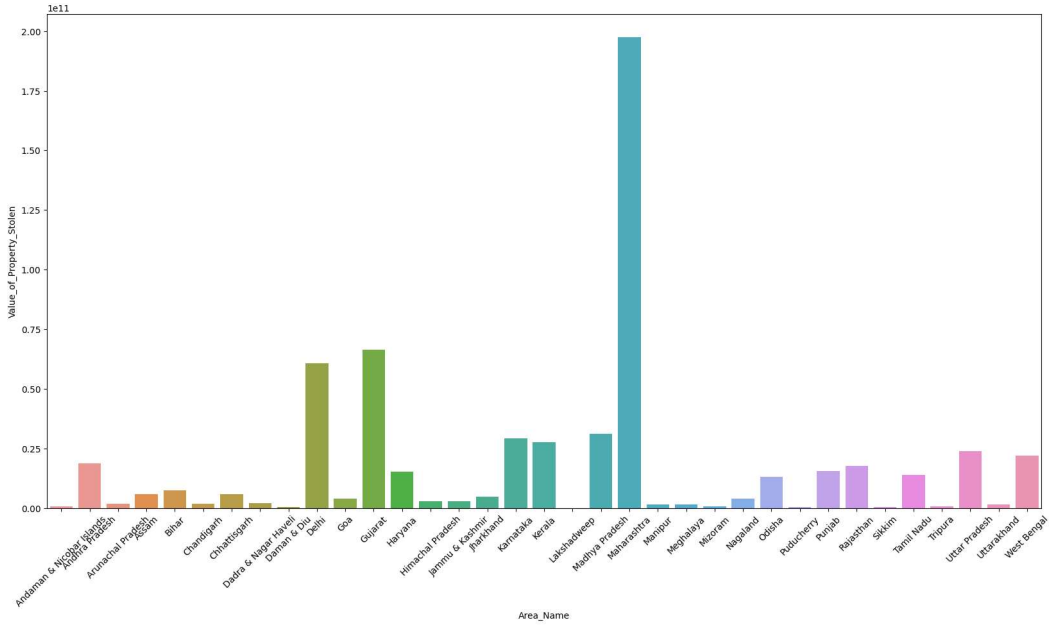

```
#Value of Property Stolen across the year of all the States

plt.figure(figsize = (20, 10))
chart=sns.barplot(x=sortbyyear.Year,y=sortbyyear.Value_of_Property_Stolen)
chart.set_xticklabels(chart.get_xticklabels(), rotation=45)
chart
```

```
<Axes: xlabel='Year', ylabel='Value_of_Property_Stolen'>
```




```
plt.figure(figsize = (20, 10))
chart=sns.barplot(x=property_bystate.Area_Name,y=property_bystate.Value_of_Property_Stolen)
chart.set_xticklabels(chart.get_xticklabels(), rotation=45)
chart
```

```
<Axes: xlabel='Area_Name', ylabel='Value_of_Property_Stolen'>
```

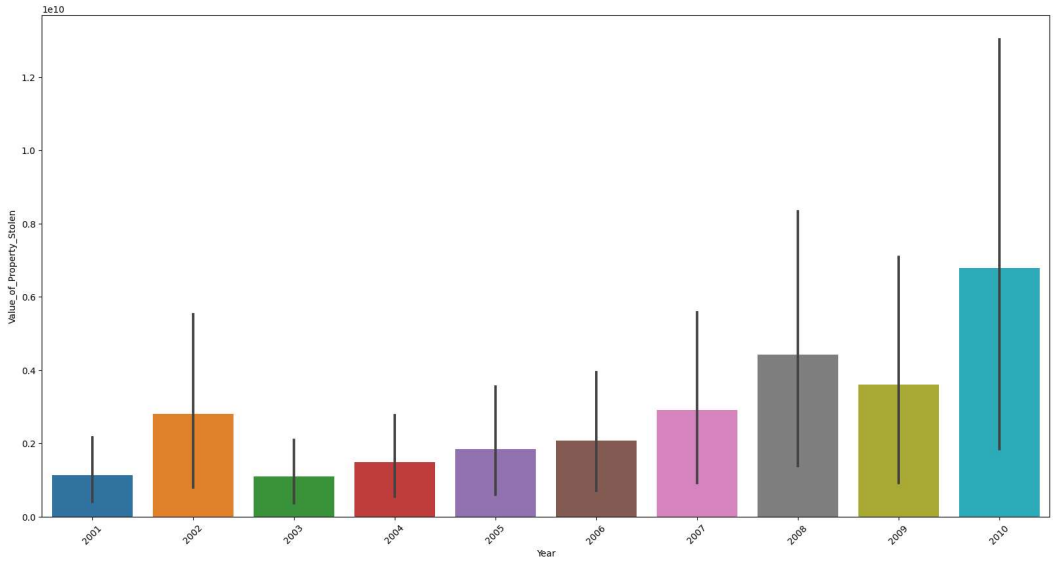


Double-click (or enter) to edit

From the Graph and the crosstab above ,we can see that Maharashtra has the most number of cases of stolen property and the value of the property stolen by a big margin So lets find out more about Maharashtra from the original dataset

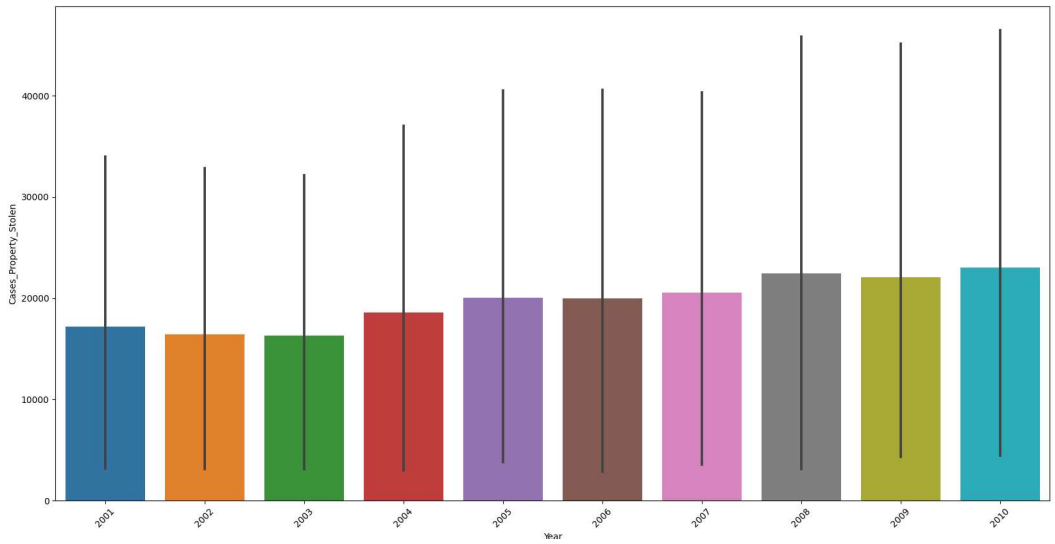
```
a=property1.groupby(['Area_Name']).get_group('Maharashtra')
plt.figure(figsize = (20, 10))
chart=sns.barplot(x=a.Year,y=a.Value_of_Property_Stolen)
chart.set_xticklabels(chart.get_xticklabels(), rotation=45)
chart
```

```
<Axes: xlabel='Year', ylabel='Value_of_Property_Stolen'>
```




```
a=property1.groupby(['Area_Name']).get_group('Maharashtra')
plt.figure(figsize = (20, 10))
chart=sns.barplot(x=a.Year,y=a.Cases_Property_Stolen)
chart.set_xticklabels(chart.get_xticklabels(), rotation=45)
chart
```

```
<Axes: xlabel='Year', ylabel='Cases_Property_Stolen'>
```



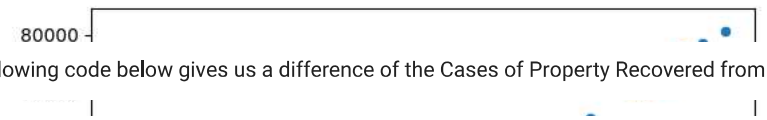

```
property.groupby(['Group_Name']).get_group('Robbery - Property')
```

	Area_Name	Year	Group_Name	Sub_Group_Name	Cases_Property_Recovered	Cases_Property_Stolen	Va
1400	Andaman & Nicobar Islands	2001	Robbery - Property	2. Robbery	2	4	
1401	Andhra Pradesh	2001	Robbery - Property	2. Robbery	293	622	
1402	Arunachal Pradesh	2001	Robbery - Property	2. Robbery	30	84	
1403	Assam	2001	Robbery - Property	2. Robbery	146	687	
1404	Bihar	2001	Robbery - Property	2. Robbery	441	2201	
...	...	...	...	...	...	...	
1745	Tamil Nadu	2010	Robbery - Property	2. Robbery	1326	1817	
1746	Tripura	2010	Robbery - Property	2. Robbery	16	63	

Scatter Plot between Cases of Property Recovered and Stolen

```
sns.scatterplot(x=property.Cases_Property_Recovered,y=property.Cases_Property_Stolen)
```

```
<Axes: xlabel='Cases_Property_Recovered', ylabel='Cases_Property_Stolen'>
```



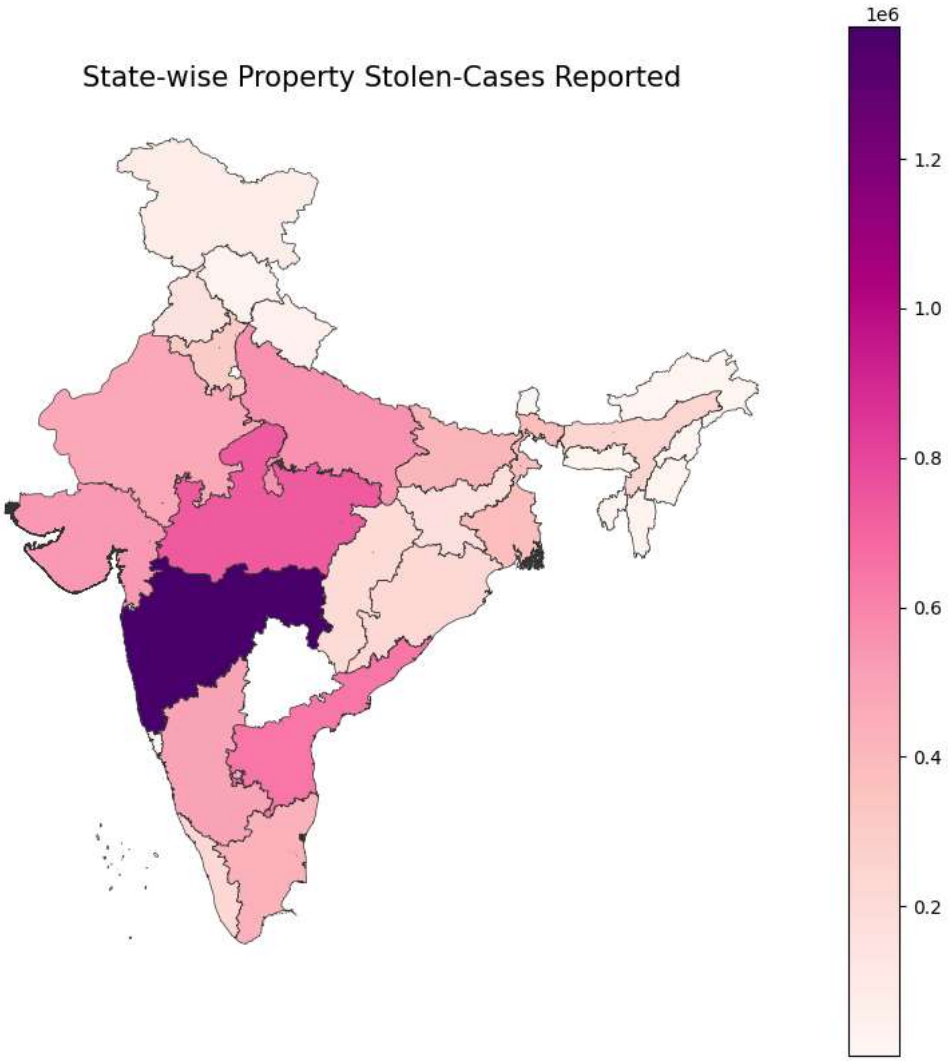
The following code below gives us a difference of the Cases of Property Recovered from Total Cases of Property Stolen

```
property_bystate['Difference']=property_bystate["Cases_Property_Stolen"]- property_bystate["Cases_Property_Recovered"]
property_bystate
```

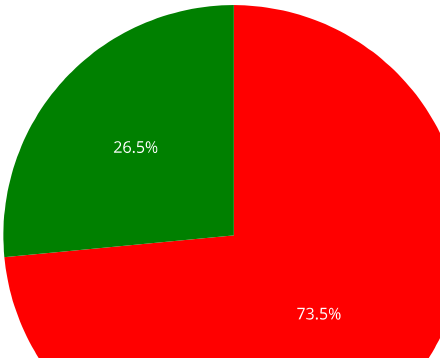
1	Andhra Pradesh	332510	642822	8320971694
2	Arunachal Pradesh	6048	16632	646754238
3	Assam	54784	245560	1521007674
4	Bihar	63876	411840	1098784766
5	Chandigarh	15188	39720	625548682
6	Chhattisgarh	68912	199712	1932428432
7	Dadra & Nagar Haveli	1170	2642	225144198
8	Daman & Diu	534	2056	81535334
9	Delhi	157858	490694	2777898238
10	Goa	5884	19788	322047336
11	Gujarat	166644	534060	6902107076
12	Haryana	133320	303336	9005055364
13	Himachal Pradesh	9798	34000	708580140
14	Jammu & Kashmir	22718	74906	1190983796
15	Jharkhand	35368	176868	688336780
16	Karnataka	160806	494968	9012218484
17	Kerala	73066	221652	3070699020
18	Lakshadweep	101	342	1591327
19	Madhya Pradesh	254106	733524	20338284748
20	Maharashtra	473186	1376814	24278687606
21	Manipur	656	11584	136829326
22	Meghalaya	3924	16724	151165908
23	Mizoram	18896	26892	345128278
24	Nagaland	2666	10814	227911296
25	Odisha	104076	224280	2761739566
26	Puducherry	7274	14236	249050464
27	Punjab	85530	151182	8637846488
28	Rajasthan	141114	469468	10094937386
29	Sikkim	966	3314	40014540
30	Tamil Nadu	342148	431864	8731172288
31	Tripura	3326	14480	88223078
32	Uttar Pradesh	171046	559970	13879052340
33	Uttarakhand	14562	41530	501626720

```
import geopandas as gpd
g7 = pd.DataFrame(property.groupby(['Area_Name'])['Cases_Property_Stolen'].sum().reset_index())
g7.columns = ['State/UT','Cases Reported']
g7.replace(to_replace='Arunachal Pradesh',value='Arunanchal Pradesh',inplace=True)

shp_gdf = gpd.read_file('/content/drive/MyDrive/Data visualozation/Crime/Indian map/India States/Indian states.shp')
```

```
merged = shp_gdf.set_index('st_nm').join(g7.set_index('State/UT'))

fig, ax = plt.subplots(1, figsize=(10, 10))
ax.axis('off')
ax.set_title('State-wise Property Stolen-Cases Reported',
             fontdict={'fontsize': '15', 'fontweight' : '3'})
fig = merged.plot(column='Cases Reported', cmap='RdPu', linewidth=0.5, ax=ax, edgecolor='0.2',legend=True)
```




```
import pandas as pd
import plotly.graph_objs as go
prop_theft_recovered = property['Cases_Property_Recovered'].sum()
prop_theft_stolen = property['Cases_Property_Stolen'].sum()

prop_group = ['Property Stolen Cases','Property Recovered Cases']
prop_vals = [prop_theft_stolen,prop_theft_recovered]

colors = ['red','green']

fig = go.Figure(data=[go.Pie(labels=prop_group, values=prop_vals,sort=False,
                             marker=dict(colors=colors),textfont_size=12)])

fig.show()
```




```
g9 = pd.DataFrame(property.groupby(['Year'])['Value_of_Property_Recovered','Value_of_Property_Stolen'].sum().reset_index())

year=['2001','2002','2003','2004','2005','2006','2007','2008','2009','2010']

fig = go.Figure(data=[
    go.Bar(name='Property Recovered', x=year, y=g9['Value_of_Property_Recovered'],
           marker_color='gold'),
    go.Bar(name='Property Stolen', x=year, y=g9['Value_of_Property_Stolen'],
          marker_color='darkblue')
])

fig.update_layout(barmode='group',xaxis_title='Year',yaxis_title='Value in Rupees',
                  title='Year-wise Value of Property Stolen and Recovered')
fig.show()
```

```
<ipython-input-26-8bae7bbb38b7>:1: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecatec
```