In [33]:

```
import os
import pathlib
```

In [34]:

```
#My first Example
print("Exploratory Data analysis la cycle")
```

```
Exploratory Data analysis la cycle
```

In [64]:

```
# importing the required library for EDA
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [7]:

```
from tkinter import Tk
from tkinter.filedialog import askopenfilename
```

In [8]:

```
Tk().withdraw()               # these three commannd will open a box on dextop, minimise this file and open your d
filename = askopenfilename()
print(filename)
```

In [65]:

```
churn=pd.read_excel("C:/Users/HP/OneDrive/Desktop/Customer Churn.xlsx")
```

In [66]:

```
churn.head(5)
```

Out[66]:

	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	Frequency of SMS	Distinct Called Numbers	Age Group	Tariff Plan	Status	Age	Custom Val
0	8	0	38	0	4370	71	5	17	3	1	1	30	197.6
1	0	0	39	0	318	5	7	4	2	1	2	25	46.0
2	10	0	37	0	2453	60	359	24	3	1	1	30	1536.5
3	10	0	38	0	4198	66	1	35	1	1	1	15	240.0
4	3	0	38	0	2393	58	2	33	1	1	1	15	145.8

In [38]:

```
churn.shape  #print the dimention of data
```

Out[38]:

```
(3150, 14)
```

In [39]:

```
#display the structure of data
churn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3150 entries, 0 to 3149
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Call  Failure            3150 non-null   int64
 1   Complains                3150 non-null   int64
 2   Subscription  Length     3150 non-null   int64
 3   Charge  Amount           3150 non-null   int64
 4   Seconds of Use           3150 non-null   int64
 5   Frequency of use         3150 non-null   int64
 6   Frequency of SMS         3150 non-null   int64
 7   Distinct Called Numbers  3150 non-null   int64
 8   Age Group                3150 non-null   int64
 9   Tariff Plan              3150 non-null   int64
 10  Status                   3150 non-null   int64
 11  Age                      3150 non-null   int64
 12  Customer Value           3150 non-null   float64
 13  Churn                    3150 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 344.7 KB
```

In [40]:

```
#rename lengthy col name
churn.rename(columns = {'Frequency of SMS':'SMS','Distinct Called Numbers':'DCN'},inplace = True)
```

In [41]:

```
churn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3150 entries, 0 to 3149
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Call  Failure         3150 non-null   int64
 1   Complains             3150 non-null   int64
 2   Subscription  Length  3150 non-null   int64
 3   Charge  Amount        3150 non-null   int64
 4   Seconds of Use        3150 non-null   int64
 5   Frequency of use      3150 non-null   int64
 6   SMS                   3150 non-null   int64
 7   DCN                   3150 non-null   int64
 8   Age Group             3150 non-null   int64
 9   Tariff Plan           3150 non-null   int64
 10  Status                3150 non-null   int64
 11  Age                   3150 non-null   int64
 12  Customer Value        3150 non-null   float64
 13  Churn                 3150 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 344.7 KB
```

In [42]:

```
churn.rename(columns = {'Subscription Length':'SL'},inplace = True)
```

In [43]:

```
#Display the type of variale
cats = list(churn.select_dtypes(include = ['object']).columns)
nums = list(churn.select_dtypes(exclude = ['object']).columns)
print(f'categorical variable:{cats}')
print(f'numerical variable:{nums}')
```

```
categorical variable:[]
numerical variable:['Call  Failure', 'Complains', 'Subscription  Length', 'Charge  Amount', 'Seconds
of Use', 'Frequency of use', 'SMS', 'DCN', 'Age Group', 'Tariff Plan', 'Status', 'Age', 'Customer Va
lue', 'Churn']
```

In [44]:

```
#discriptive analytics of data set for each column(eg mean, standard deviation, max,min)
churn.describe()
```

Out[44]:

	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	SMS	DCN	Age Grou
count	3150.000000	3150.000000	3150.000000	3150.000000	3150.000000	3150.000000	3150.000000	3150.000000	3150.00000
mean	7.627937	0.076508	32.541905	0.942857	4472.459683	69.460635	73.174921	23.509841	2.82603
std	7.263886	0.265851	8.573482	1.521072	4197.908687	57.413308	112.237560	17.217337	0.89255
min	0.000000	0.000000	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.00000
25%	1.000000	0.000000	30.000000	0.000000	1391.250000	27.000000	6.000000	10.000000	2.00000
50%	6.000000	0.000000	35.000000	0.000000	2990.000000	54.000000	21.000000	21.000000	3.00000
75%	12.000000	0.000000	38.000000	1.000000	6478.250000	95.000000	87.000000	34.000000	3.00000
max	36.000000	1.000000	47.000000	10.000000	17090.000000	255.000000	522.000000	97.000000	5.00000

In [45]:

```
#to convert certain numerical into categorical variable
churn['Age']=pd.Categorical(churn['Age'])
churn['complains']=pd.Categorical(churn['Complains'])
```

In [46]:

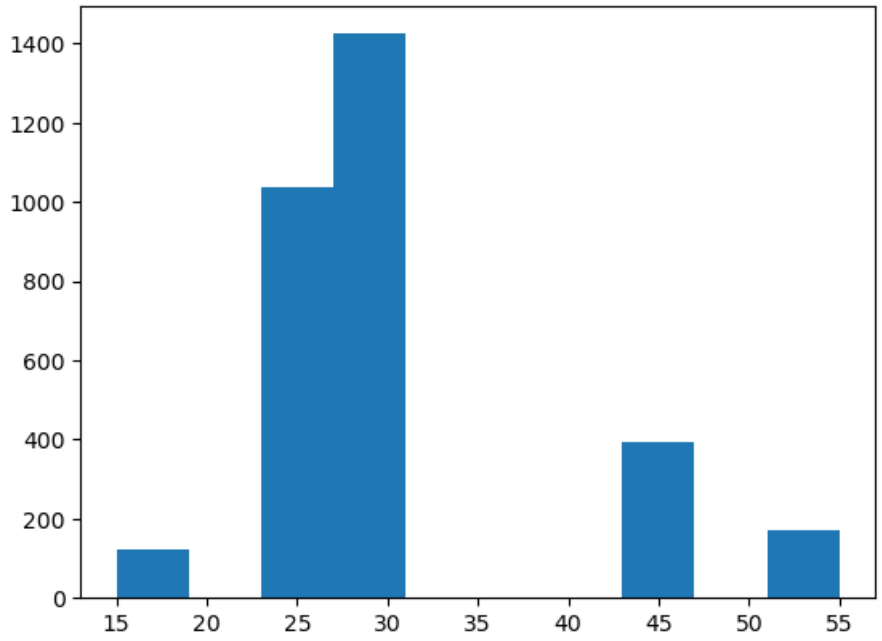
```
#visualisation of data

import matplotlib.pyplot as plt
import seaborn as sn
#%matplotlib inline
```

In [47]:

```
plt.hist(churn['Age'])
```

Out[47]:

```
(array([ 123.,    0., 1037., 1425.,    0.,    0.,    0.,  395.,    0.,
         170.]),
 array([15., 19., 23., 27., 31., 35., 39., 43., 47., 51., 55.]),
 <BarContainer object of 10 artists>)
```

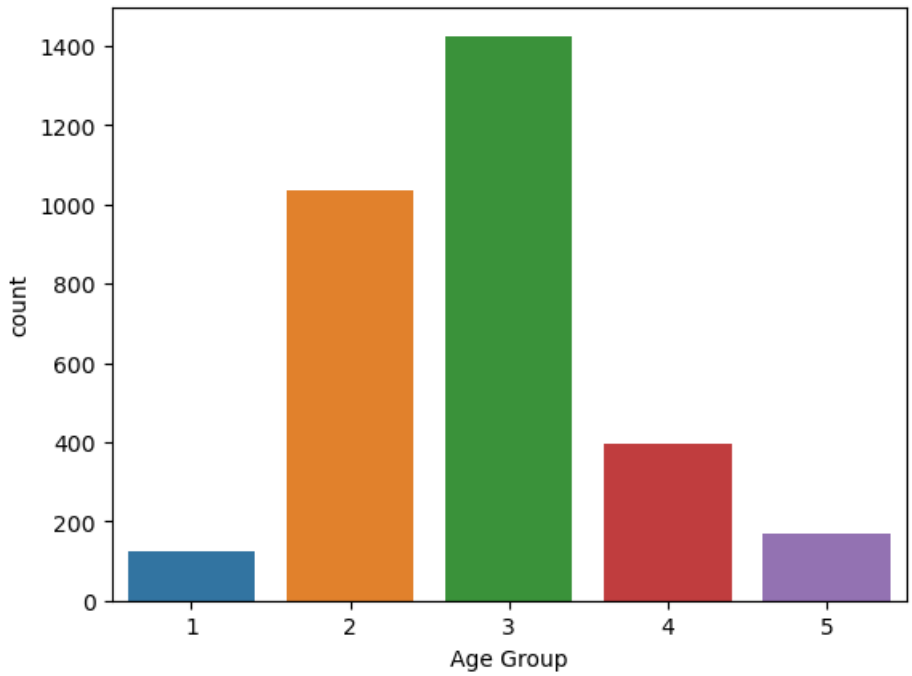


In [48]:

```
#count plot for categorical variable
sn.countplot(x = 'Age Group',data=churn)
```

Out[48]:

```
<Axes: xlabel='Age Group', ylabel='count'>
```

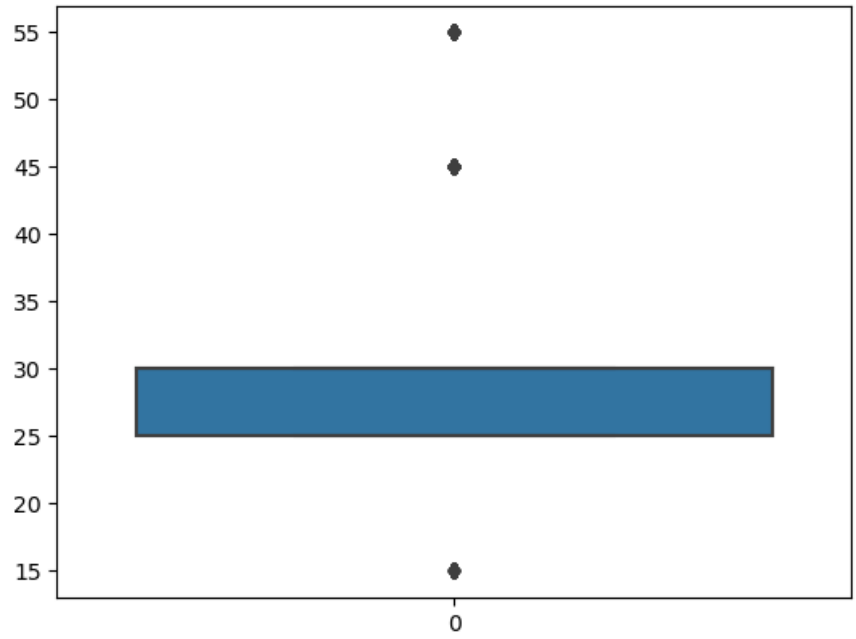


In [49]:

```
#box plot for continuous variable
#should be drown to identify the precence of outliers
sn.boxplot(churn['Age'])
```

Out[49]:

```
<Axes: >
```



In [50]:

```
#dealing withh missing values
churn.isnull().sum()    # since there is no mising value, we ignore this
```

Out[50]:

```
Call  Failure           0
Complains               0
Subscription  Length    0
Charge  Amount          0
Seconds of Use          0
Frequency of use        0
SMS                     0
DCN                     0
Age Group               0
Tariff Plan             0
Status                  0
Age                     0
Customer Value          0
Churn                   0
complains               0
dtype: int64
```

In [51]:

```
#standardize the dataset for numerical attributes
nums = list(churn.select_dtypes(exclude=['object']).columns)
nums   # to show output
```

Out[51]:

```
['Call  Failure',
 'Complains',
 'Subscription  Length',
 'Charge  Amount',
 'Seconds of Use',
 'Frequency of use',
 'SMS',
 'DCN',
 'Age Group',
 'Tariff Plan',
 'Status',
 'Age',
 'Customer Value',
 'Churn',
 'complains']
```

In [52]:

```
#scale the data
from sklearn import preprocessing
min_max_scaler=preprocessing.MinMaxScaler()
churn[['Call  Failure',
 'Complains',
 'Subscription  Length',
 'Charge  Amount',
 'Seconds of Use',
 'Frequency of use',
 'SMS',
 'DCN',
 'Age Group',
 'Tariff Plan',
 'Status',
 'Age',
 'Customer Value',
 'Churn',
 'complains']]
churn.head()
```

Out[52]:

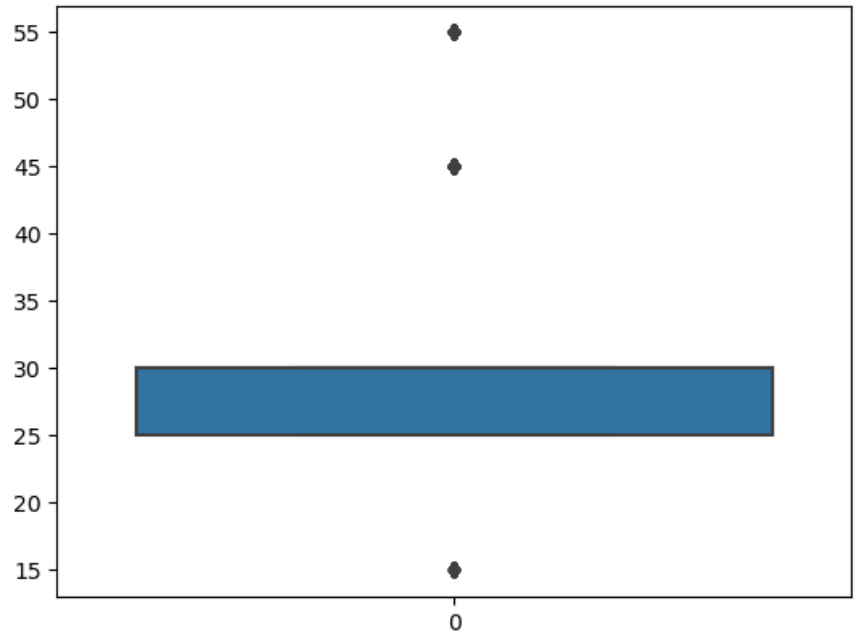
	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	SMS	DCN	Age Group	Tariff Plan	Status	Age	Customer Value	Churn
0	8	0	38	0	4370	71	5	17	3	1	1	30	197.640	0
1	0	0	39	0	318	5	7	4	2	1	2	25	46.035	0
2	10	0	37	0	2453	60	359	24	3	1	1	30	1536.520	0
3	10	0	38	0	4198	66	1	35	1	1	1	15	240.020	0
4	3	0	38	0	2393	58	2	33	1	1	1	15	145.805	0

In [53]:

```
sn.boxplot(churn['Age'])
```

Out[53]:

<Axes: >



In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [54]:

```
#class2
```

In [55]:

```
churn.head()
```

Out[55]:

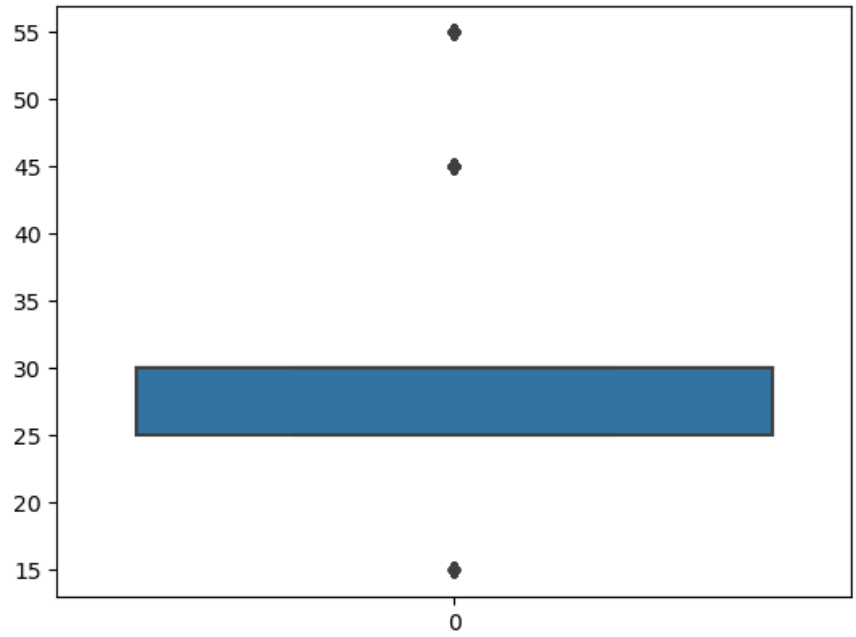
	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	SMS	DCN	Age Group	Tariff Plan	Status	Age	Customer Value	Churn
0	8	0	38	0	4370	71	5	17	3	1	1	30	197.640	0
1	0	0	39	0	318	5	7	4	2	1	2	25	46.035	0
2	10	0	37	0	2453	60	359	24	3	1	1	30	1536.520	0
3	10	0	38	0	4198	66	1	35	1	1	1	15	240.020	0
4	3	0	38	0	2393	58	2	33	1	1	1	15	145.805	0

In [56]:

```
sn.boxplot(churn['Age'])
```

Out[56]:

<Axes: >



In [ ]:

```

```

In [68]:

```
import pandas as pd

# Assuming 'churn' is the DataFrame or variable containing the 'Age' column
Q1 = churn['Age'].quantile(0.25)
Q3 = churn['Age'].quantile(0.75)
IQR = Q3 - Q1
print(Q1)
print(Q3)
print(IQR)
Lower_Whisker = Q1 - (1.5 * IQR)
Upper_Whisker = Q3 + (1.5 * IQR)
print(Lower_Whisker, Upper_Whisker)
```

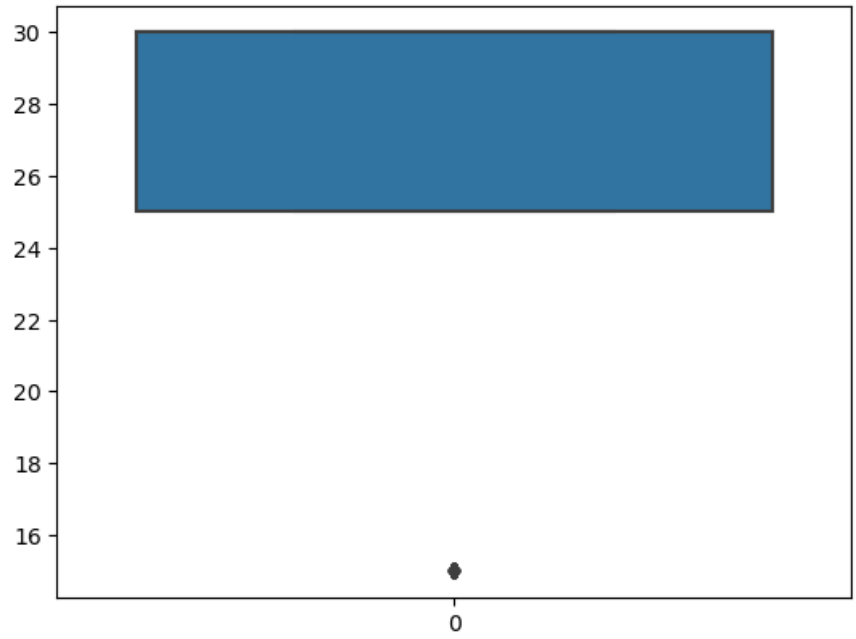
```
25.0
30.0
5.0
17.5 37.5
```

In [98]:

```
#Remove he outliers
churn = churn[churn['Age']<Upper_Whisker]
churn.head()
sn.boxplot(churn['Age'])
```

Out[98]:

```
<Axes: >
```

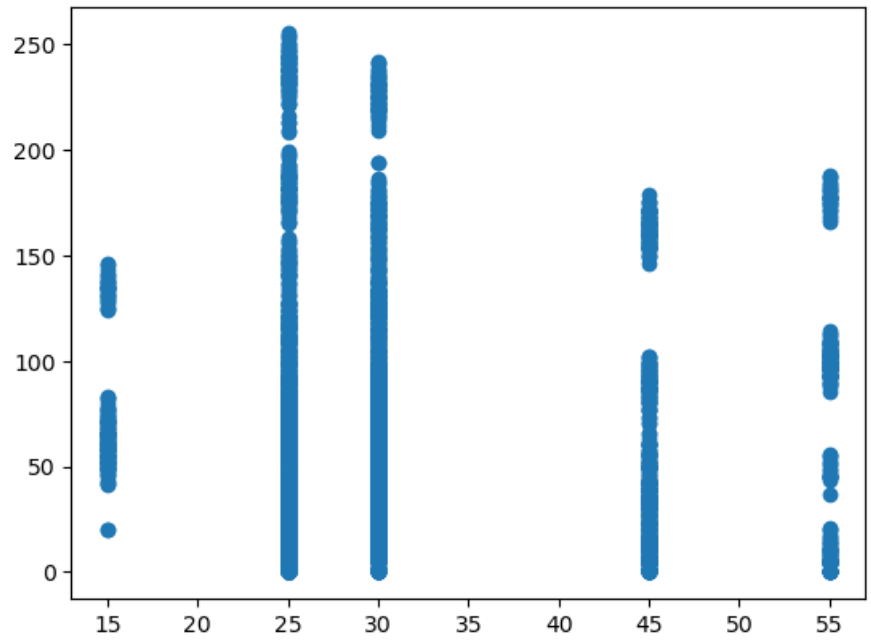


In [72]:

```
#importing the required libraries for EDA
#scatter plot
plt.scatter(x=churn['Age'],y=churn['Frequency of use'])
```

Out[72]:

```
<matplotlib.collections.PathCollection at 0x1dfbee8a110>
```

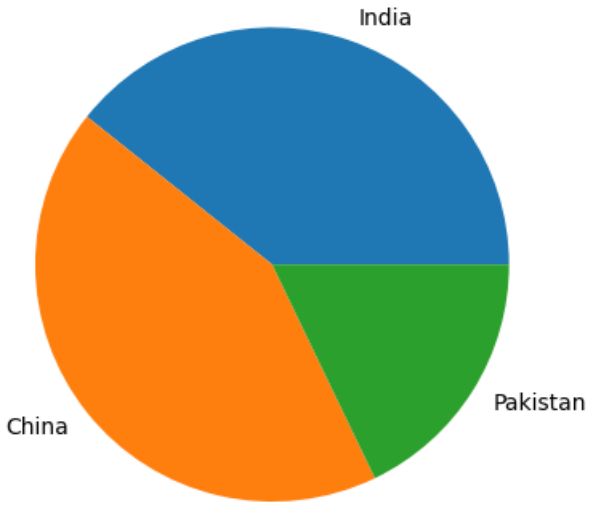


In [75]:

```
#piechart
Country =['India','China','Pakistan']
Population = [55,60,25]
plt.pie(Population,labels=Country)
```

Out[75]:

```
([<matplotlib.patches.Wedge at 0x1dfbeddd540>,
  <matplotlib.patches.Wedge at 0x1dfbeddd450>,
  <matplotlib.patches.Wedge at 0x1dfbeddde10>],
 [Text(0.363306995924972, 1.0382716536205603, 'India'),
  Text(-0.8600146582346299, -0.6858387475358717, 'China'),
  Text(0.9313966113234436, -0.5852352966245341, 'Pakistan')])
```

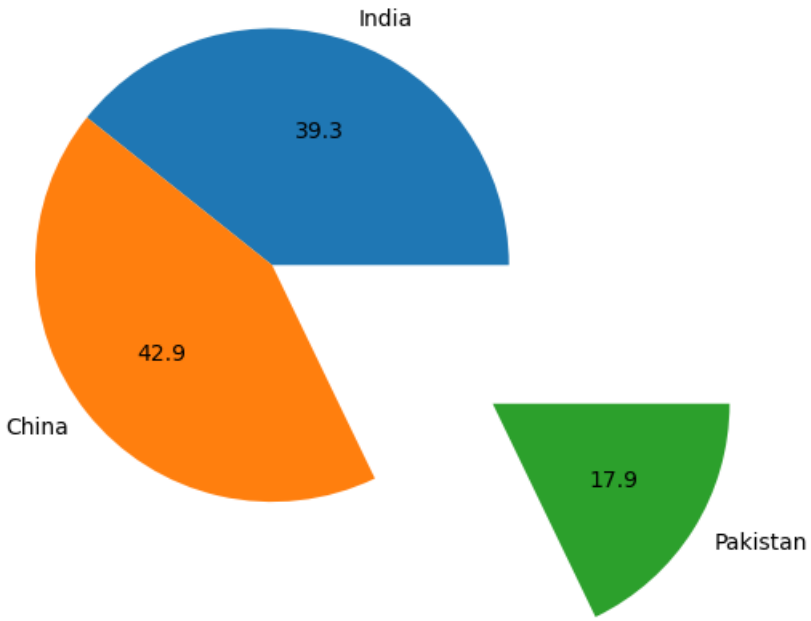


In [78]:

```
#pie with explosion
slices = (0,0,1.1)
plt.pie(Population,labels=Country,explode=slices,autopct='%0.1f')
```

Out[78]:

```
([<matplotlib.patches.Wedge at 0x1dfbed87520>,
  <matplotlib.patches.Wedge at 0x1dfbed87430>,
  <matplotlib.patches.Wedge at 0x1dfbefe8250>],
 [Text(0.363306995924972, 1.0382716536205603, 'India'),
  Text(-0.8600146582346299, -0.6858387475358717, 'China'),
  Text(1.8627932226468873, -1.1704705932490682, 'Pakistan')],
 [Text(0.198167452322712, 0.5663299928839419, '39.3'),
  Text(-0.4690989044916162, -0.37409386229229363, '42.9'),
  Text(1.4394311265907764, -0.9044545493288254, '17.9')])
```

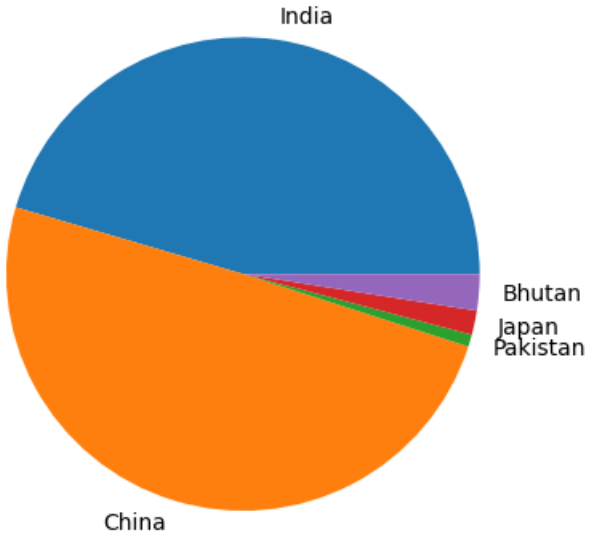


In [88]:

```
#piechart
Country =['India','China','Pakistan','Japan','Bhutan']
Population = [55,60,1,2,3]
plt.pie(Population,labels=Country)
```

Out[88]:

```
([<matplotlib.patches.Wedge at 0x1dfc1ebc2b0>,
  <matplotlib.patches.Wedge at 0x1dfc1ebc1c0>,
  <matplotlib.patches.Wedge at 0x1dfc1e8f2b0>,
  <matplotlib.patches.Wedge at 0x1dfc1ebcee0>,
  <matplotlib.patches.Wedge at 0x1dfc1ebd360>],
 [Text(0.15654627576372776, 1.0888035927312634, 'India'),
  Text(-0.32358078419504555, -1.051330336335692, 'China'),
  Text(1.0554422783122464, -0.30990578754043135, 'Pakistan'),
  Text(1.0763566163639837, -0.22684010758566564, 'Japan'),
  Text(1.09666485728739, -0.08559317023471415, 'Bhutan')])
```

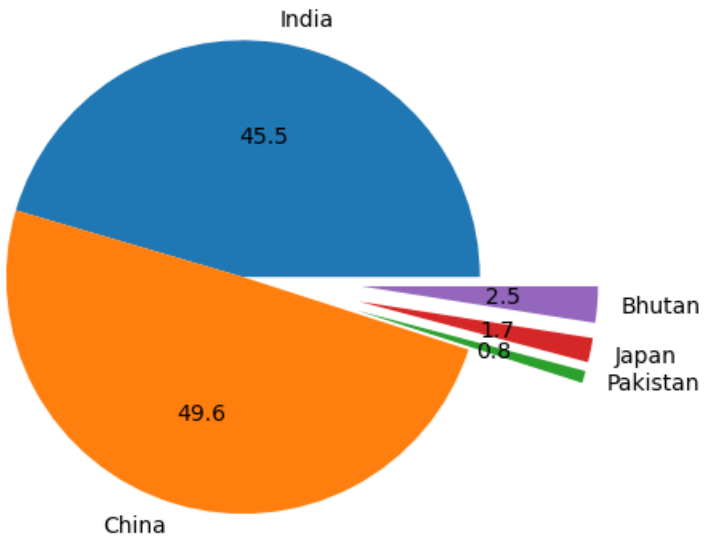


In [91]:

```
#pie with explosion
slices = (0,0,0.5,0.5,0.5)
plt.pie(Population,labels=Country,explode=slices,autopct='%0.1f')
```

Out[91]:

```
([<matplotlib.patches.Wedge at 0x1dfc212c3d0>,
  <matplotlib.patches.Wedge at 0x1dfc212c2e0>,
  <matplotlib.patches.Wedge at 0x1dfc212d0c0>,
  <matplotlib.patches.Wedge at 0x1dfc212d750>,
  <matplotlib.patches.Wedge at 0x1dfc212dd80>],
 [Text(0.15654627576372776, 1.0888035927312634, 'India'),
  Text(-0.32358078419504555, -1.051330336335692, 'China'),
  Text(1.5351887684541765, -0.4507720546042638, 'Pakistan'),
  Text(1.5656096238021582, -0.32994924739733184, 'Japan'),
  Text(1.5951488833271128, -0.12449915670503875, 'Bhutan')],
 [Text(0.08538887768930604, 0.5938928687625072, '45.5'),
  Text(-0.1764986095609339, -0.5734529107285593, '49.6'),
  Text(1.0554422783122461, -0.3099057875404313, '0.8'),
  Text(1.0763566163639835, -0.22684010758566564, '1.7'),
  Text(1.0966648572873898, -0.08559317023471413, '2.5')])
```

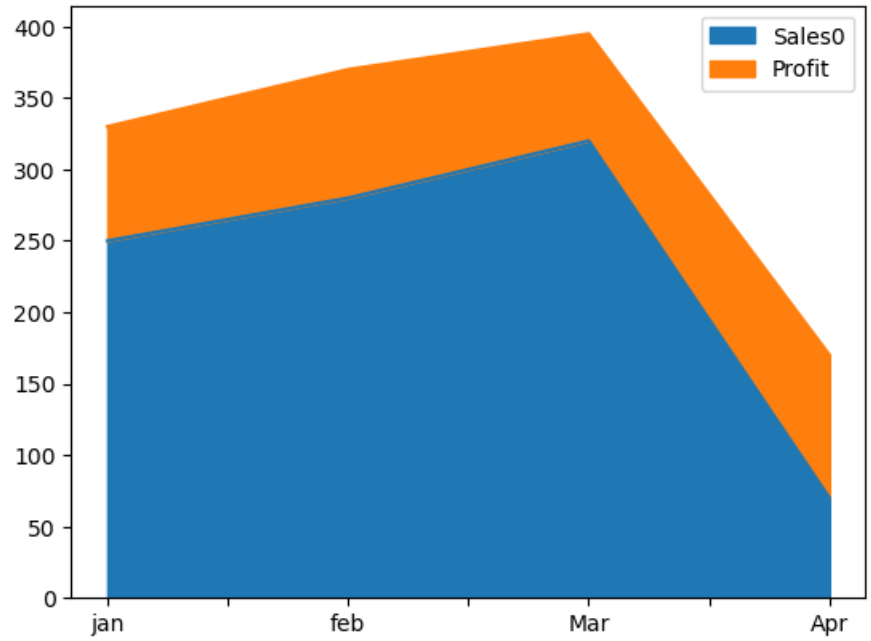


In [96]:

```
#Area chart - continuousvariable can be staked one upon the other
data={'Sales0':[250,280,320,70],'Profit':[80,90,75,100]};
months = ("jan","feb","Mar","Apr");
df = pd.DataFrame(data,index=months);
df.plot.area()
```

Out[96]:

<Axes: >

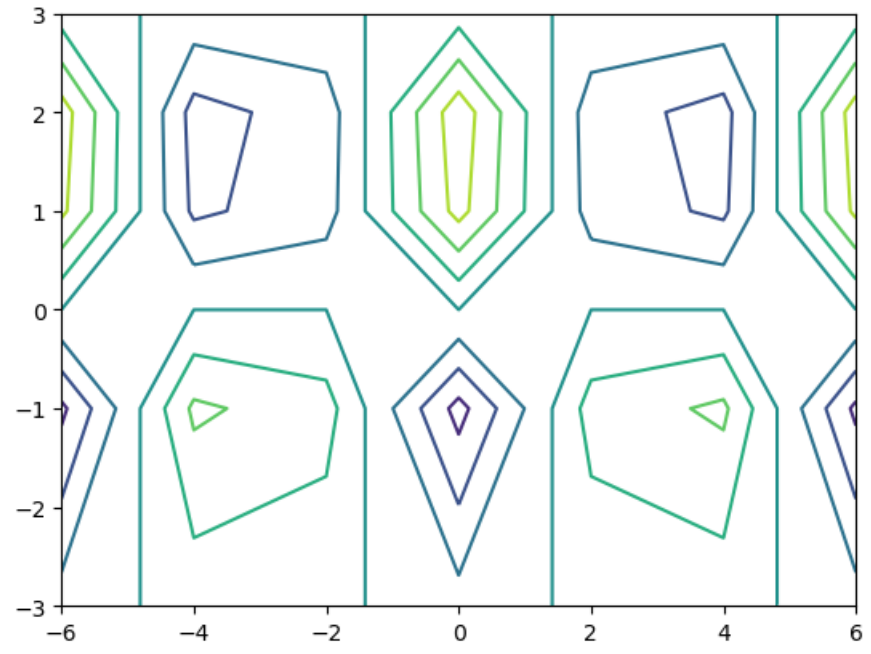


In [105]:

```
#counter plot its a three dimensionnal plot
x=(-6,-4,-2,0,2,4,6)
y=(-3,-3,-1,0,1,2,3)
xvalue,yvalue=np.meshgrid(x,y)
zvalue=np.cos(xvalue)*np.sin(yvalue)
plt.contour(xvalue,yvalue,zvalue)
```

Out[105]:

<matplotlib.contour.QuadContourSet at 0x1dfc24a8850>

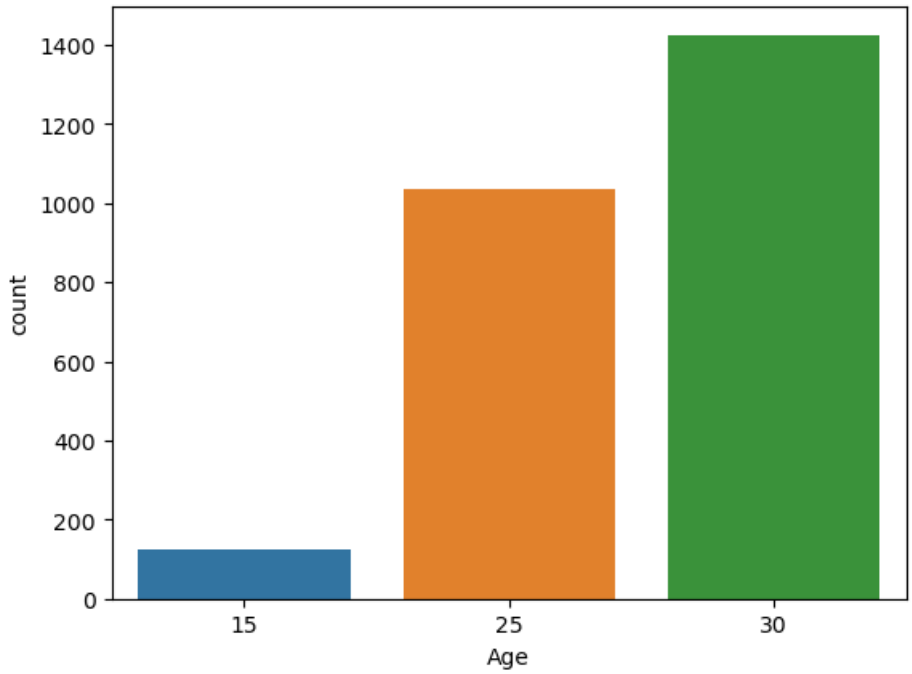


In [111]:

```
#seaborn
sns.countplot(x='Age',data=churn)
```

Out[111]:

```
<Axes: xlabel='Age', ylabel='count'>
```

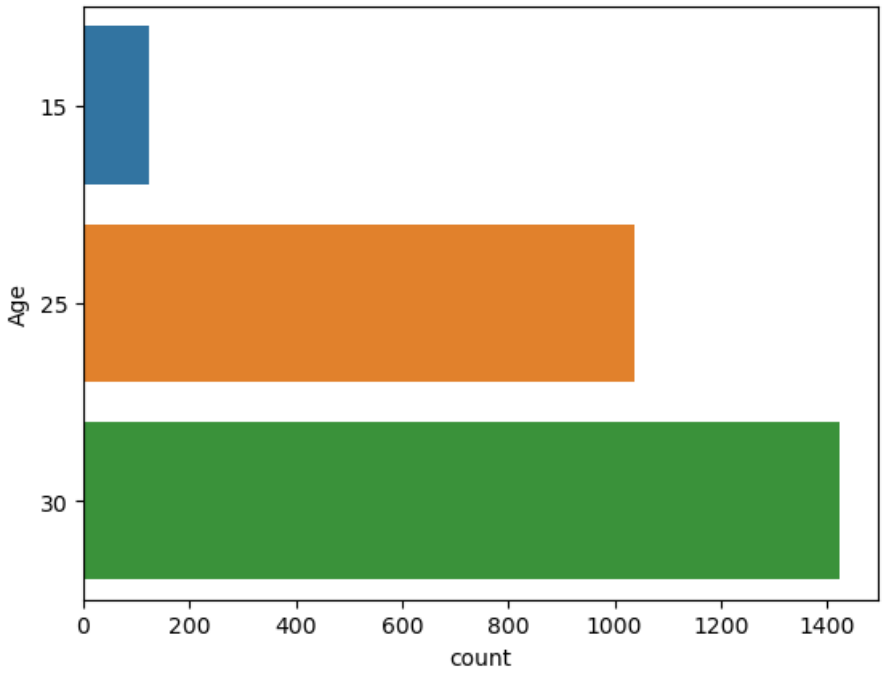


In [118]:

```
#seaborn
sns.countplot(y='Age',data=churn)
```

Out[118]:

```
<Axes: xlabel='count', ylabel='Age'>
```



In [113]:

```
churn.head()
```

Out[113]:

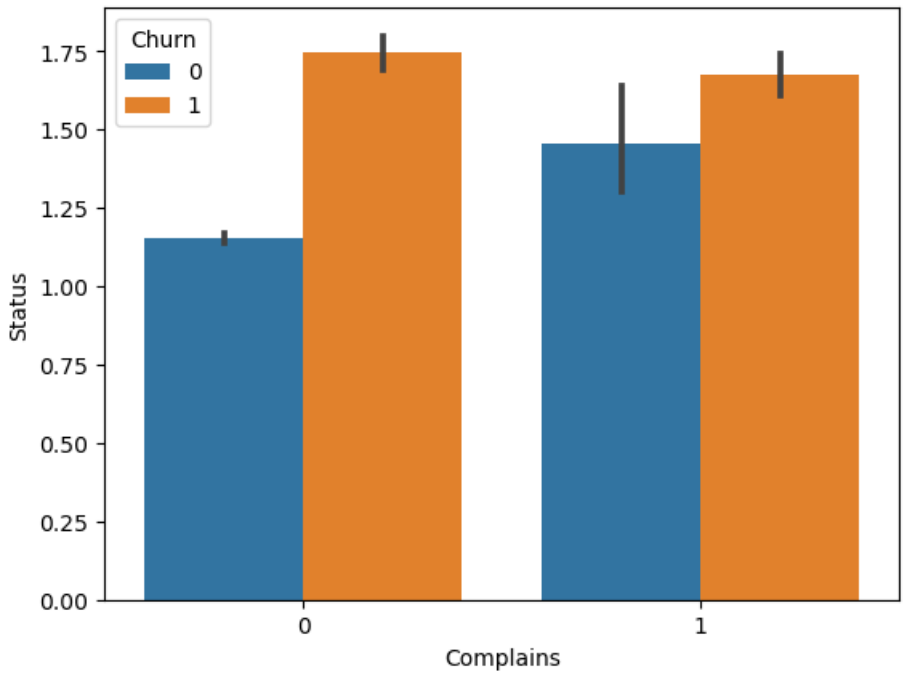
	Call Failure	Complains	Subscription Length	Charge Amount	Seconds of Use	Frequency of use	Frequency of SMS	Distinct Called Numbers	Age Group	Tariff Plan	Status	Age	Custom Val
0	8	0	38	0	4370	71	5	17	3	1	1	30	197.6
1	0	0	39	0	318	5	7	4	2	1	2	25	46.0
2	10	0	37	0	2453	60	359	24	3	1	1	30	1536.5
3	10	0	38	0	4198	66	1	35	1	1	1	15	240.0
4	3	0	38	0	2393	58	2	33	1	1	1	15	145.8

In [124]:

```
import seaborn as sns
sns.barplot(x='Complains',y='Status',hue="Churn",data=churn)
```

Out[124]:

```
<Axes: xlabel='Complains', ylabel='Status'>
```

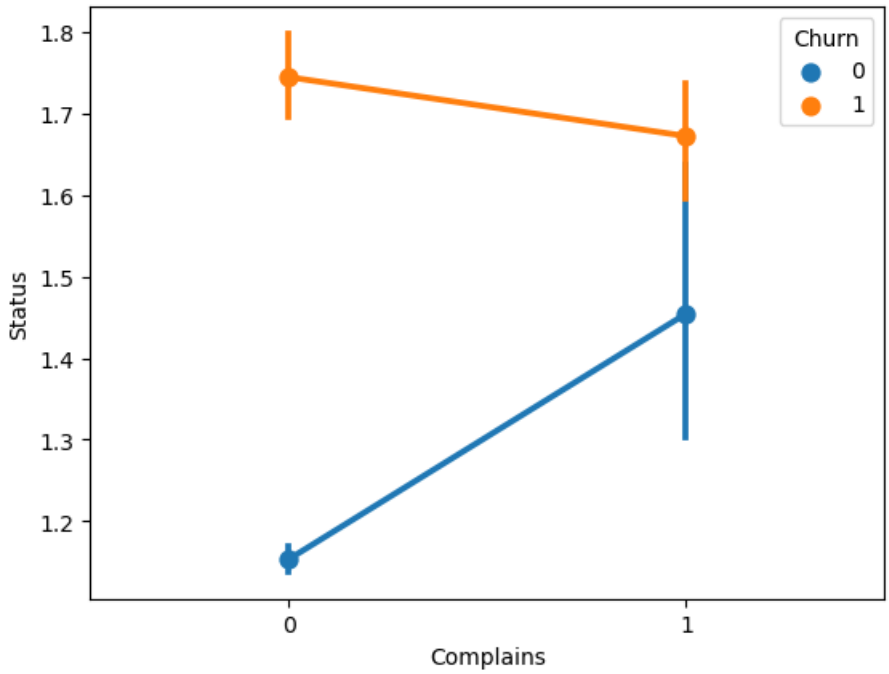


In [131]:

```
#for point plot one should be categorical and other is numerical. here "Complains" is categorical. try for Age alsc
sns.pointplot(x='Complains',y='Status',hue="Churn",data=churn)
```

Out[131]:

```
<Axes: xlabel='Complains', ylabel='Status'>
```

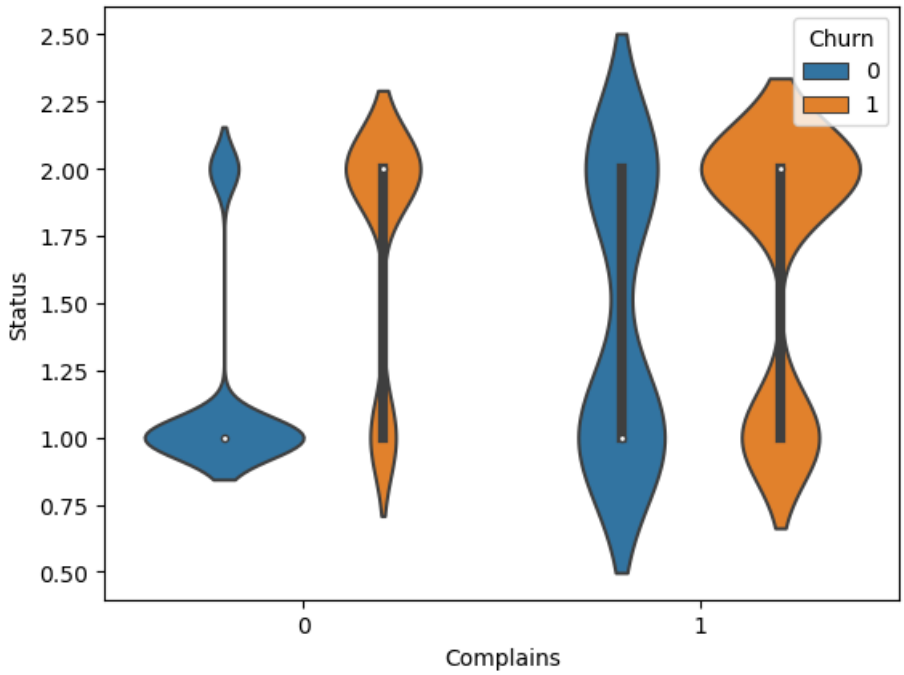


In [129]:

```
#violin plot
sns.violinplot(x='Complains',y='Status',hue="Churn",data=churn)
```

Out[129]:

```
<Axes: xlabel='Complains', ylabel='Status'>
```



###

In [ ]:

```
```