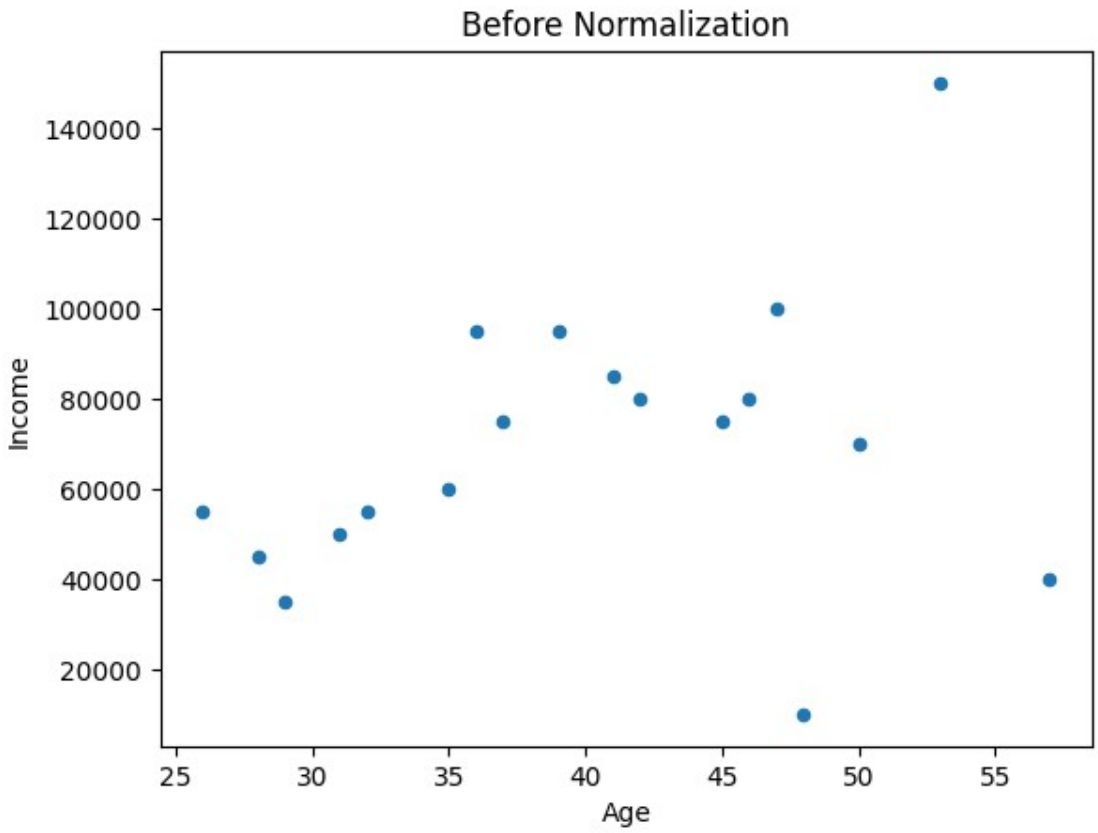
```
import pandas as pd
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
import numpy as np

per=pd.read_csv('/content/drive/MyDrive/Project/
person.csv',index_col=0,na_values=['??','????'])
per.dropna(axis=0,inplace=True)

per.columns

Index(['Age', 'Gender', 'Marital Status', 'Income'], dtype='object')

per.head()

               Age  Gender Marital Status  Income
City
New York      32.0    Male         Single   55000
Toronto       45.0  Female        Married   75000
Paris         28.0    Male         Single   45000
London        31.0    Male         Single   50000
Los Angeles   57.0  Female       Divorced   40000

per.plot.scatter(x='Age',y='Income',title='Before Normalization')

/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/
core.py:1114: UserWarning: No data for colormapping provided via 'c'.
Parameters 'cmap' will be ignored
  scatter = ax.scatter(

<Axes: title={'center': 'Before Normalization'}, xlabel='Age',
ylabel='Income'>
```




```
per_numeric=per.select_dtypes(include='number')
per_numeric.columns
```

```
Index(['Age', 'Income'], dtype='object')
```

```
per.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18 entries, New York to Tokyo
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             18 non-null     float64
 1   Gender          18 non-null     object
 2   Marital Status  18 non-null     object
 3   Income          18 non-null     int64
dtypes: float64(1), int64(1), object(2)
memory usage: 720.0+ bytes
```

```
def max_abs_scaling(per):
 per_scaled=per.copy()
 for col in per_scaled.columns[:1]:
  per_scaled[col]=per_scaled[col]/per_scaled[col].abs().max()
 return per_scaled
```

```
per_cars_scaled=max_abs_scaling(per)
per_cars_scaled.head()
```

```
                  Age  Gender Marital Status  Income
City
New York      0.561404    Male         Single   55000
Toronto       0.789474  Female        Married   75000
Paris         0.491228    Male         Single   45000
London        0.543860    Male         Single   50000
Los Angeles   1.000000  Female       Divorced   40000
```

```
abs_scaler=MaxAbsScaler()
```

```
per1=per[['Age','Income']]
per1.columns
```

```
Index(['Age', 'Income'], dtype='object')
```

```
abs_scaler.fit(per1)
```

```
MaxAbsScaler()
```

```
abs_scaler.fit(per1)
```

```
MaxAbsScaler()
```

```
abs_scaler.max_abs_
```

```
array([5.7e+01, 1.5e+05])
```

```
scaled_data=abs_scaler.transform(per1)
```

```
scaled_data.dtype
```

```
dtype('float64')
```

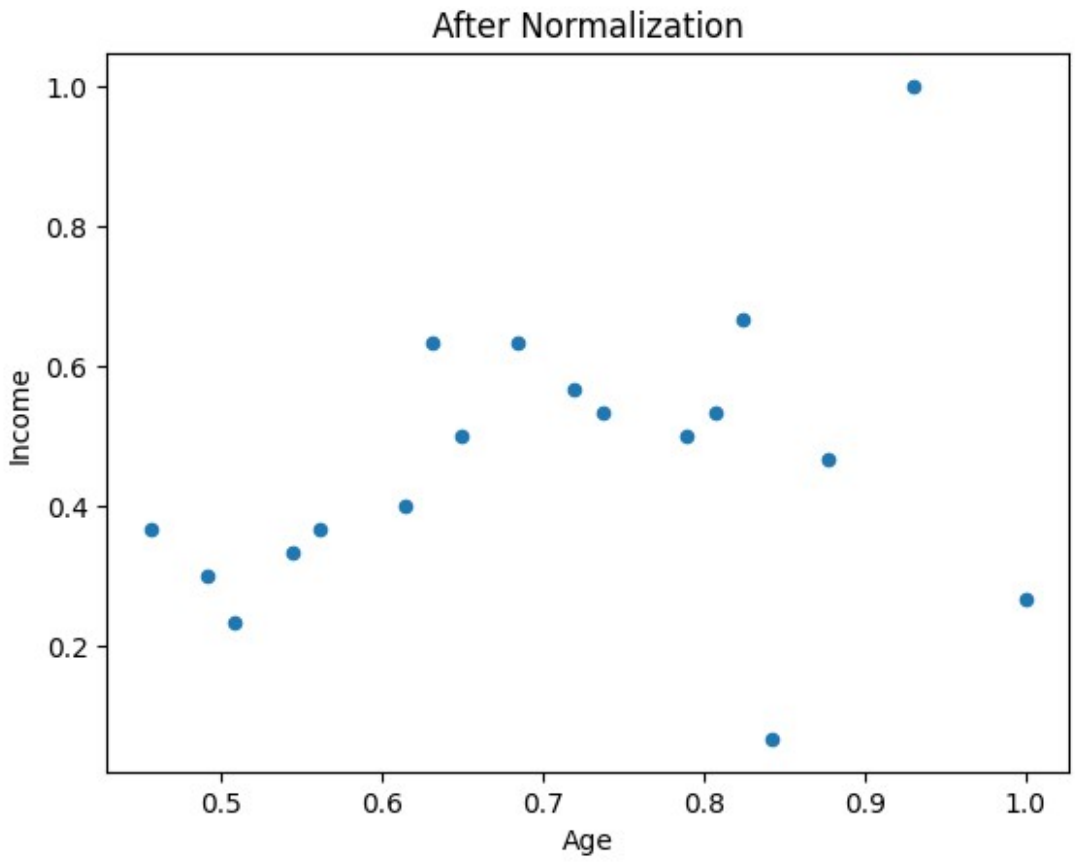
```
per_scaled_cars=pd.DataFrame(scaled_data, columns=per1.columns)
per_scaled_cars.head()
```

```
        Age    Income
0  0.561404  0.366667
1  0.789474  0.500000
2  0.491228  0.300000
3  0.543860  0.333333
4  1.000000  0.266667
```

```
per_scaled_cars.plot.scatter(x='Age',y='Income',title="After
Normalization")
```

```
/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/
core.py:1114: UserWarning: No data for colormapping provided via 'c'.
Parameters 'cmap' will be ignored
  scatter = ax.scatter(
```

```
<Axes: title={'center': 'After Normalization'}, xlabel='Age',
ylabel='Income'>
```



2)Min-Max Normalization

```
def min_max_scaling(df):
 df_scaled=df.copy()
 for col in df_scaled.columns:
  df_scaled[col]=(df_scaled[col]-
df_scaled[col].min())/(df_scaled[col].max()-df_scaled[col].min())
 return df_scaled

per_cars_scaled=min_max_scaling(per[['Age','Income']])
per_cars_scaled.head()
```

```
                   Age    Income
City
New York      0.193548  0.321429
Toronto       0.612903  0.464286
Paris         0.064516  0.250000
London        0.161290  0.285714
Los Angeles   1.000000  0.214286
```

```
MMscaler=MinMaxScaler()
per_norm=pd.DataFrame(MMscaler.fit_transform(per1),columns=per1.column
s)
per_norm.head()
```

```
        Age    Income
0  0.193548  0.321429
1  0.612903  0.464286
2  0.064516  0.250000
3  0.161290  0.285714
4  1.000000  0.214286
```

```
MMscaler.data_min_
```

```
array([   26., 10000.])
```

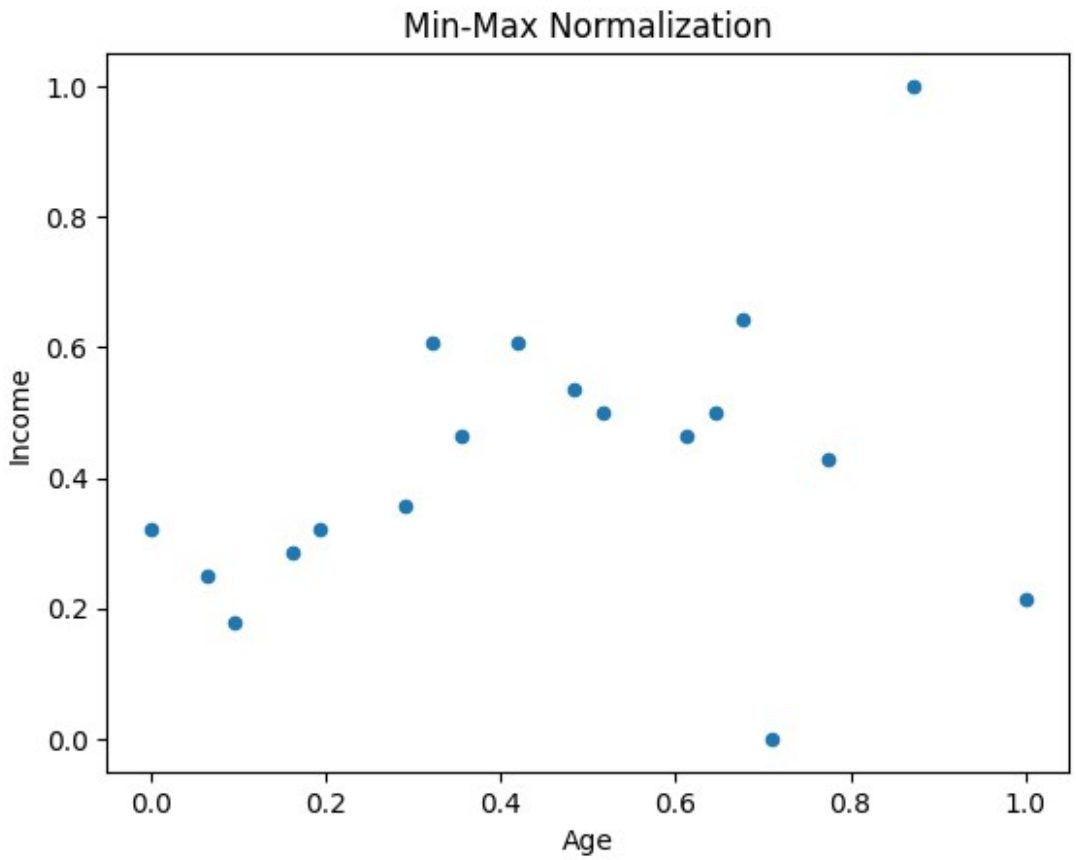
```
MMscaler.data_max_
```

```
array([5.7e+01, 1.5e+05])
```

```
per_norm.plot.scatter(x='Age',y='Income',title='Min-Max
Normalization')
```

```
/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/
core.py:1114: UserWarning: No data for colormapping provided via 'c'.
Parameters 'cmap' will be ignored
  scatter = ax.scatter(
```

```
<Axes: title={'center': 'Min-Max Normalization'}, xlabel='Age',
ylabel='Income'>
```



3)z-score method

```
def z_norm(df):
 df_scaled=df.copy()
 for col in df_scaled.columns:
  df_scaled[col]=(df_scaled[col]-
df_scaled[col].mean())/(df_scaled[col].std())
 return df_scaled

per_scaled_cars=z_norm(per[['Age','Income']])
per_scaled_cars.head()
```

```
                  Age    Income
City
New York     -0.895794 -0.475105
Toronto       0.539930  0.170321
Paris        -1.337555 -0.797818
London       -1.006234 -0.636462
Los Angeles   1.865214 -0.959174
```

```
stdscaler=StandardScaler()
stdscaler_data=stdscaler.fit_transform(per1)
stdscaler_df=pd.DataFrame(stdscaler_data,columns=per1.columns)
stdscaler_df.head()
```

```
        Age    Income
0 -0.921764 -0.488879
1  0.555584  0.175259
2 -1.376333 -0.820948
3 -1.035406 -0.654914
4  1.919290 -0.986982
```

```
stdscaler.scale_
```

```
array([8.79955105e+00, 3.01142372e+04])
```

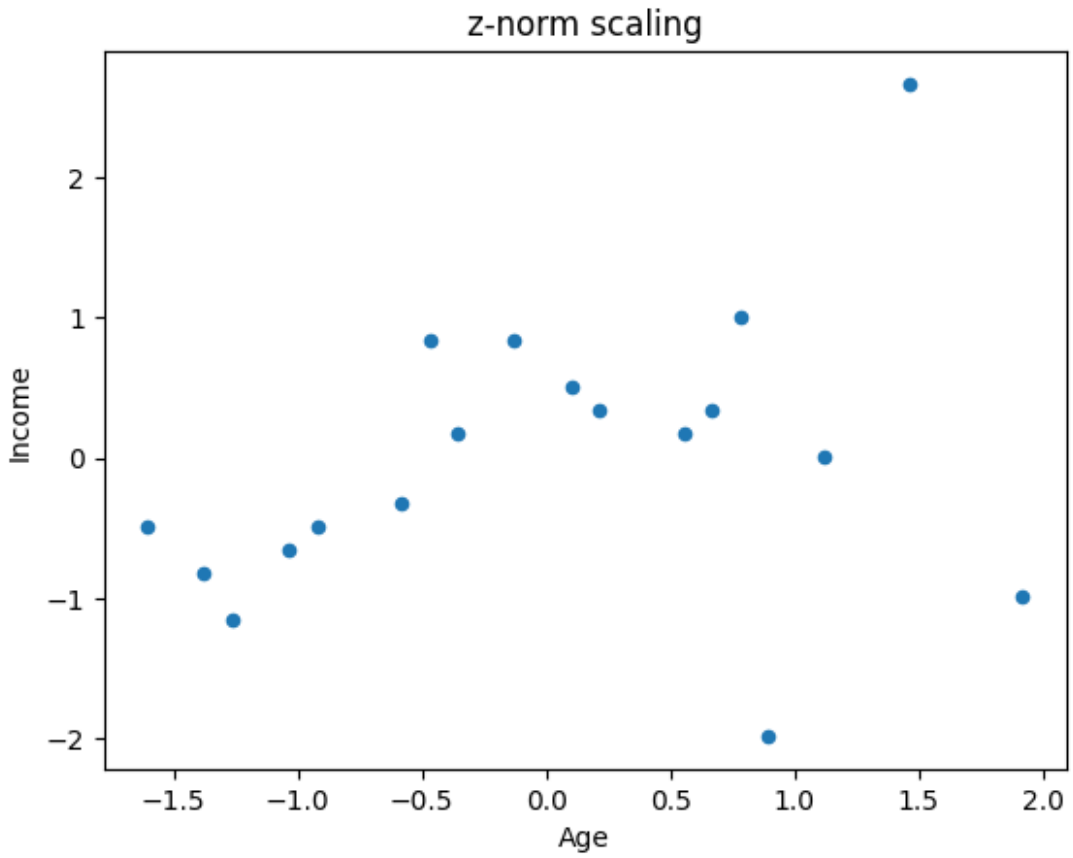
```
stdscaler.mean_
```

```
array([4.01111111e+01, 6.97222222e+04])
```

```
stdscaler_df.plot.scatter(x='Age',y='Income',title="z-norm scaling")
```

```
/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/
core.py:1114: UserWarning: No data for colormapping provided via 'c'.
Parameters 'cmap' will be ignored
  scatter = ax.scatter(
```

```
<Axes: title={'center': 'z-norm scaling'}, xlabel='Age',
ylabel='Income'>
```



4)Robust Scaling

```
def robust_scaling(df):
 df_scaled=df.copy()
 for col in df_scaled.columns:
  df_scaled[col]=(df_scaled[col]-
df_scaled[col].median())/(df_scaled[col].quantile(0.75)-
df_scaled[col].quantile(0.25))
 return df_scaled

df_scaled=robust_scaling(per1)
df_scaled.head()
```

```
                   Age    Income
City
New York     -0.571429 -0.538462
Toronto       0.357143  0.076923
Paris        -0.857143 -0.846154
London       -0.642857 -0.692308
Los Angeles   1.214286 -1.000000
```

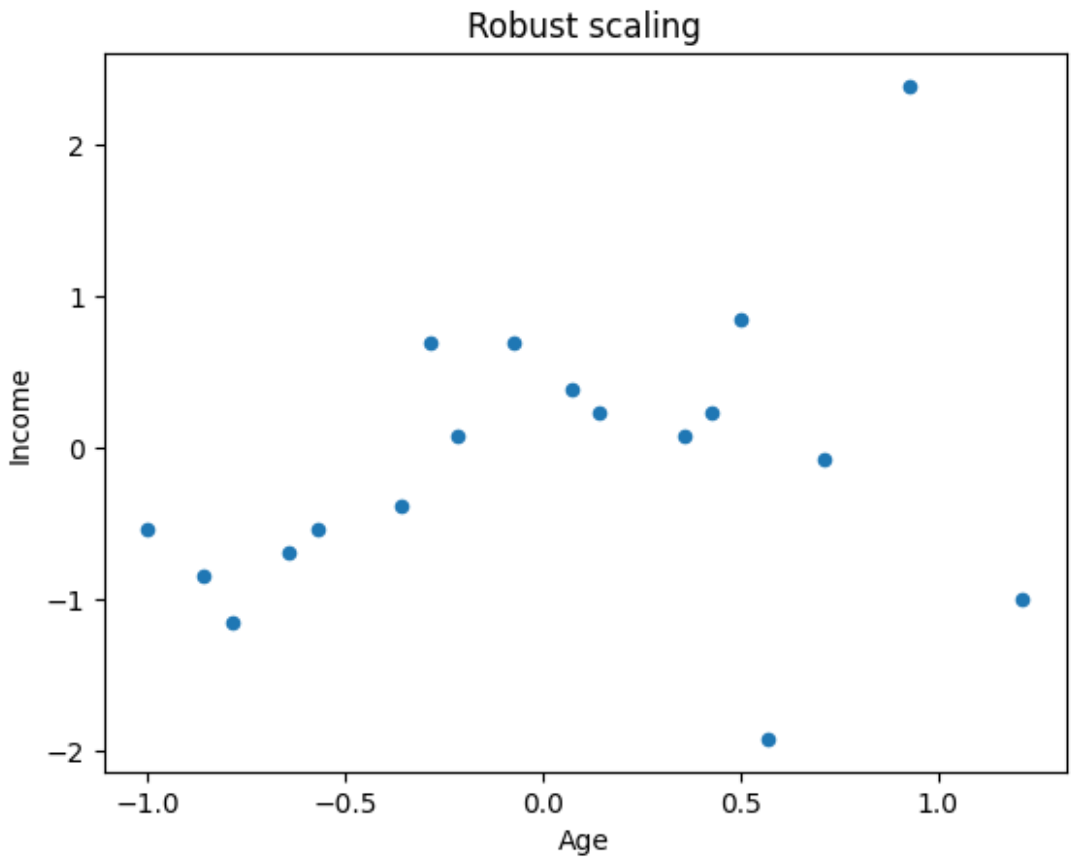
```
rob_scale=RobustScaler()
rob_scaledata=rob_scale.fit_transform(per1)
rob_scaled_df=pd.DataFrame(rob_scaledata, columns=per1.columns)
rob_scaled_df.head()
```

```
        Age    Income
0 -0.571429 -0.538462
1  0.357143  0.076923
2 -0.857143 -0.846154
3 -0.642857 -0.692308
4  1.214286 -1.000000
```

```
rob_scaled_df.plot.scatter(x='Age',y='Income',title='Robust scaling')
```

```
/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/
core.py:1114: UserWarning: No data for colormapping provided via 'c'.
Parameters 'cmap' will be ignored
  scatter = ax.scatter(
```

```
<Axes: title={'center': 'Robust scaling'}, xlabel='Age',
ylabel='Income'>
```



5)Binning

```
min_age=per1['Age'].min()
max_age=per1['Age'].max()
print(min_age)
print(max_age)
```

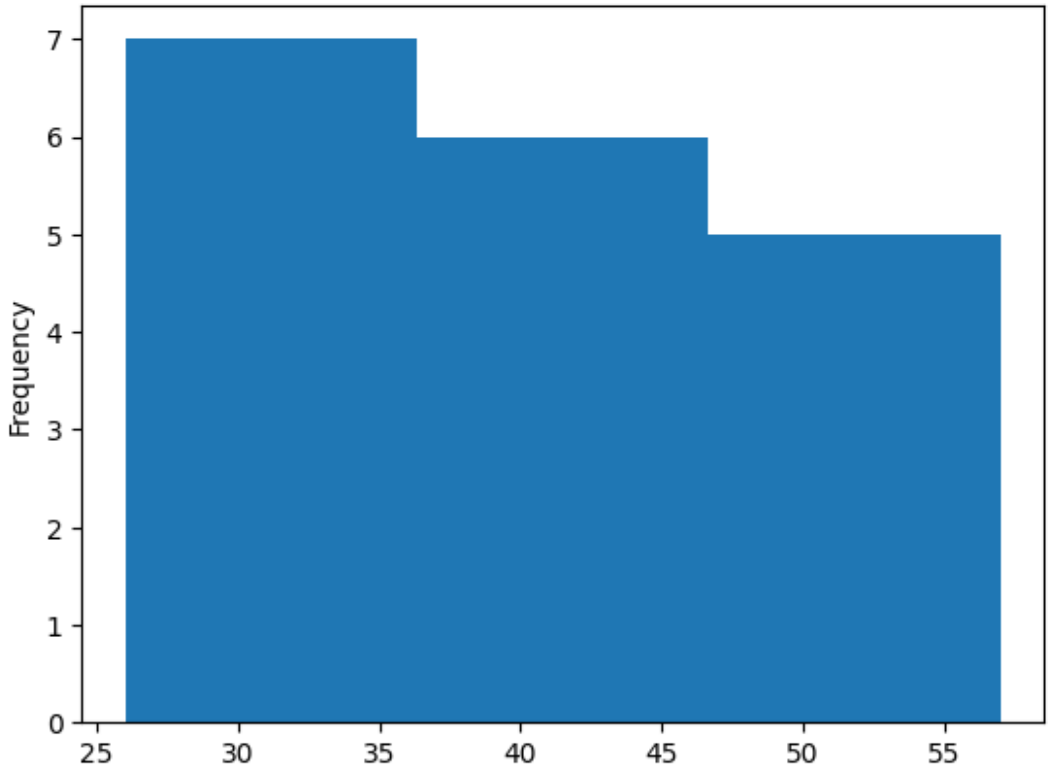
```
26.0
57.0
```

```
np.linspace(1,10, 4)
```

```
array([ 1.,  4.,  7., 10.])
```

```
#use linspace() to get the 4 bins
bins=np.linspace(min_age,max_age,4)
bins
labels =['young','middle','senior']

per1['Age_categ']=pd.cut(per1['Age'],bins=bins,labels=labels,include_l
owest=True)
per1.columns
```

```
<ipython-input-112-11e00e9e9e2f>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy

per1['Age_categ']=pd.cut(per1['Age'],bins=bins,labels=labels,include_l
owest=True)

Index(['Age', 'Income', 'Age_categ'], dtype='object')

per1['Age'].plot.hist(bins=3)

<Axes: ylabel='Frequency'>
```




```
per1['Age_categ_freq']=pd.qcut(per1['Age'],q=3,labels=labels,precision
=1)
per1.columns

Index(['Age', 'Income', 'Age_categ', 'Age_categ_freq'],
dtype='object')

per1.head()
```

City	Age	Income	Age_categ	Age_categ_freq
New York	32.0	55000	young	young
Toronto	45.0	75000	middle	middle
Paris	28.0	45000	young	young
London	31.0	50000	young	young
Los Angeles	57.0	40000	senior	senior