```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

```
data = pd.read_csv('segmentation data.csv')
```

```
data.head()
```

	ID	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
0	100000001	0	0	67	2	124670	1	2
1	100000002	1	1	22	1	150773	1	2
2	100000003	0	0	49	1	89210	0	0
3	100000004	0	0	45	1	171565	1	1
4	100000005	0	0	53	1	149031	1	1

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID               2000 non-null   int64
 1   Sex              2000 non-null   int64
 2   Marital status   2000 non-null   int64
 3   Age              2000 non-null   int64
 4   Education        2000 non-null   int64
 5   Income           2000 non-null   int64
 6   Occupation       2000 non-null   int64
 7   Settlement size  2000 non-null   int64
dtypes: int64(8)
memory usage: 125.1 KB
```

```
#lets checkk the null values
data.isna().sum()
```

```
ID                0
Sex               0
Marital status    0
Age               0
Education         0
Income            0
Occupation        0
Settlement size   0
dtype: int64
```

```
#lets check duplicates
data.duplicated().sum()
```

```
0
```

```
#lets drop the id column
data.drop('ID',inplace=True,axis = 1)
```

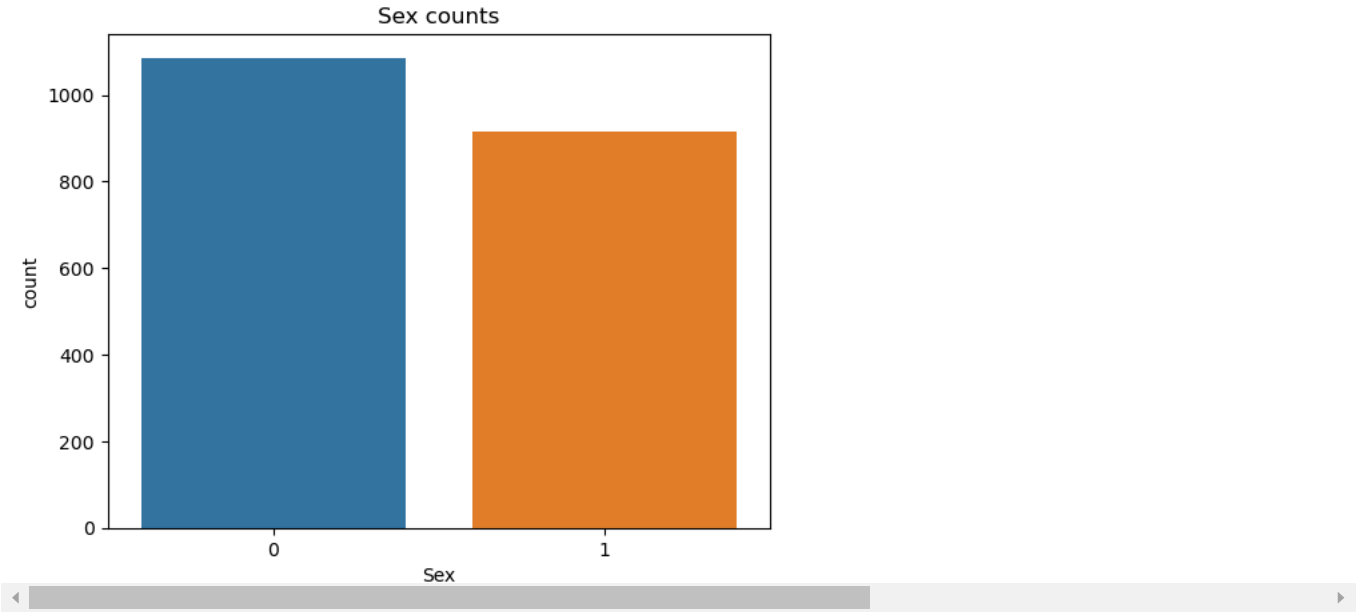
```
data.head()
```

	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
0	0	0	67	2	124670	1	2
1	1	1	22	1	150773	1	2
2	0	0	49	1	89210	0	0
3	0	0	45	1	171565	1	1
4	0	0	53	1	149031	1	1

```
sex = data['Sex'].value_counts()
print(sex)
sns.countplot(x='Sex',data=data)
```

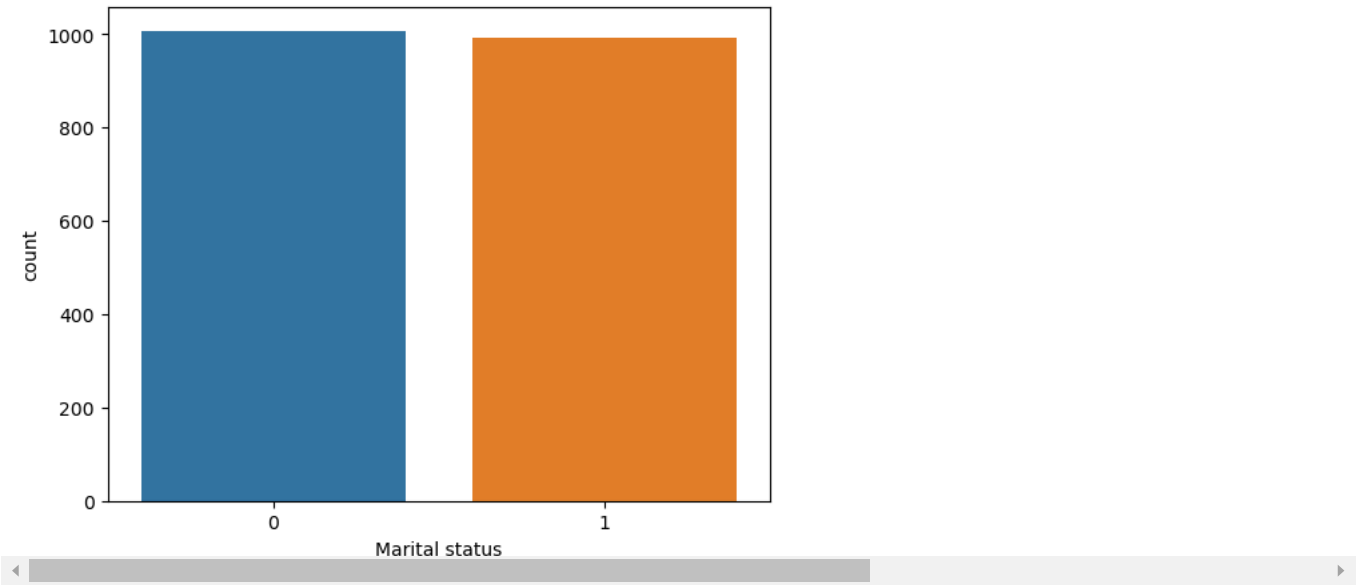
```
plt.title('Sex counts')
plt.show()
```

```
0    1086
1     914
Name: Sex, dtype: int64
```



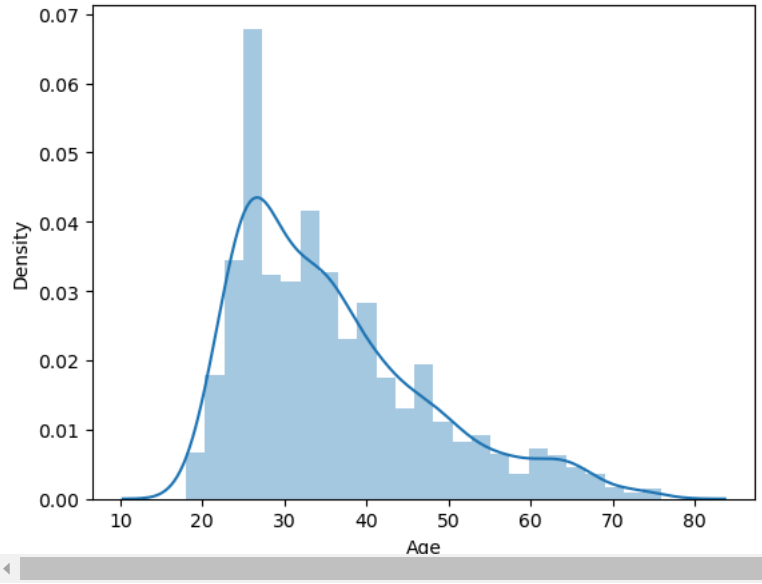

```
MS = data['Marital status'].value_counts()
print(MS)
sns.countplot(x='Marital status',data=data)
```

```
0    1007
1     993
Name: Marital status, dtype: int64
<Axes: xlabel='Marital status', ylabel='count'>
```



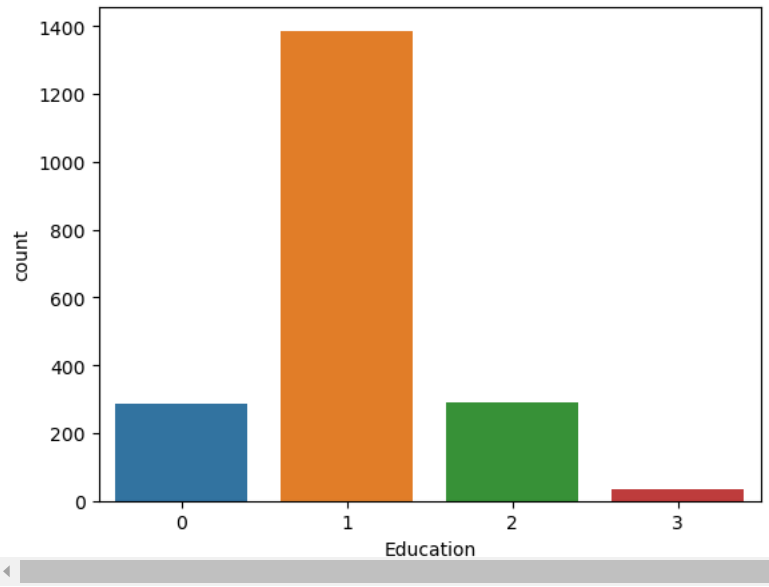

```
#lets calculate the mean age
mean_age = data['Age'].mean()
print('Mean Age: ',mean_age)
sns.distplot(data['Age'])
```

```
Mean Age:  35.909
<Axes: xlabel='Age', ylabel='Density'>
```



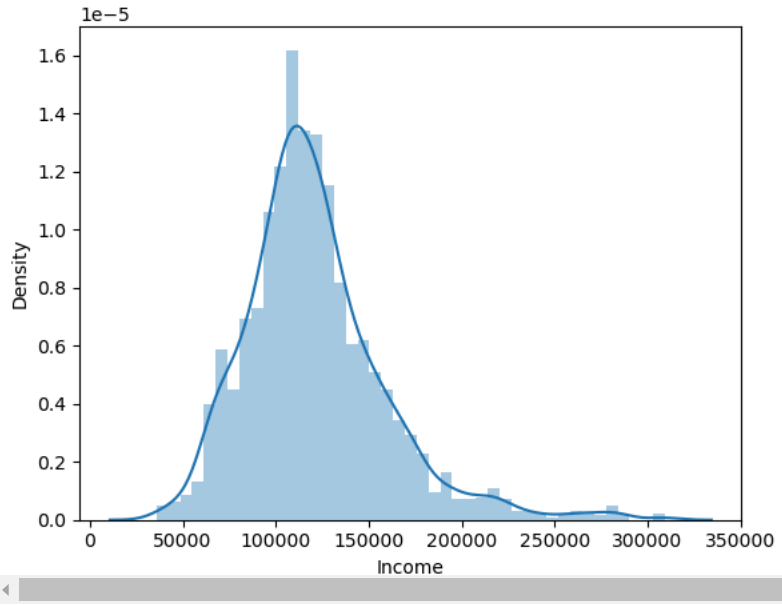

```
Education = data['Education'].value_counts()
print(Education)
sns.countplot(x='Education',data=data)
```

```
1    1386
2     291
0     287
3      36
Name: Education, dtype: int64
<Axes: xlabel='Education', ylabel='count'>
```



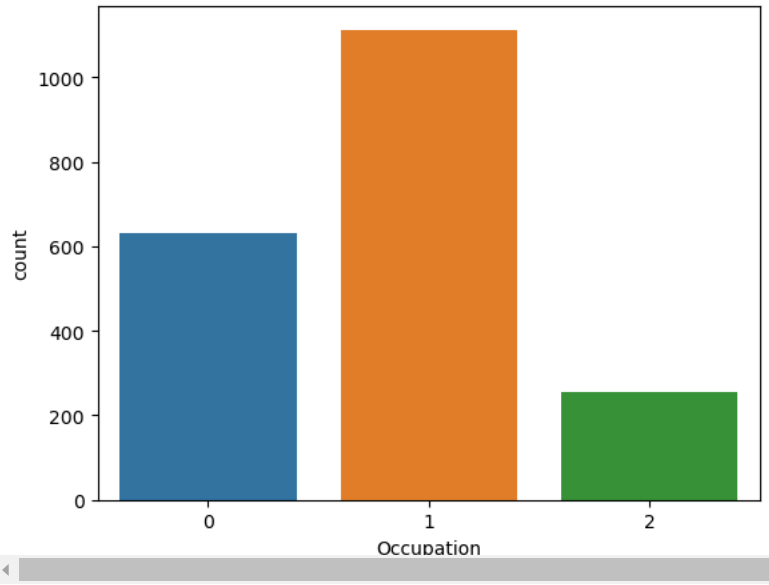

```
#lets see the mean income
mean_income = data['Income'].mean()
print('Mean Income: ',mean_income)
sns.distplot(data['Income'])
```

```
Mean Income:  120954.419
<Axes: xlabel='Income', ylabel='Density'>
```



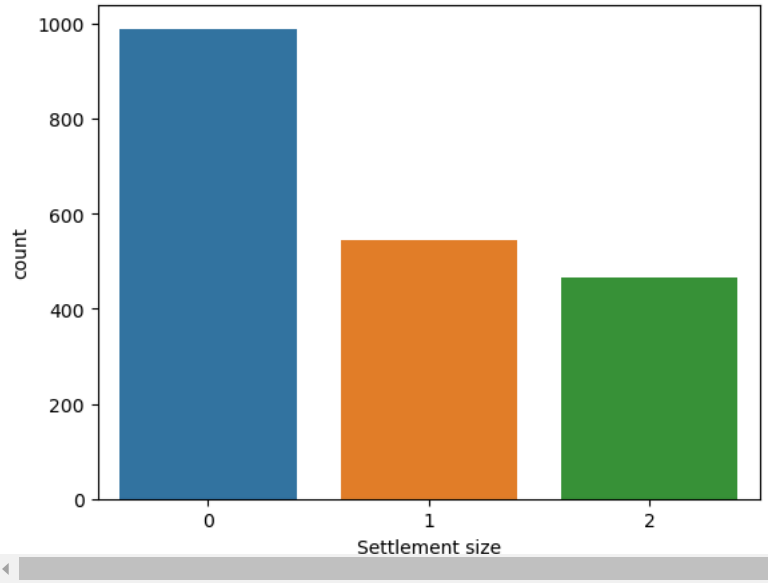

```
occ = data['Occupation'].value_counts()
print(occ)
sns.countplot(x='Occupation',data=data)
```

```
1    1113
0     633
2     254
Name: Occupation, dtype: int64
<Axes: xlabel='Occupation', ylabel='count'>
```



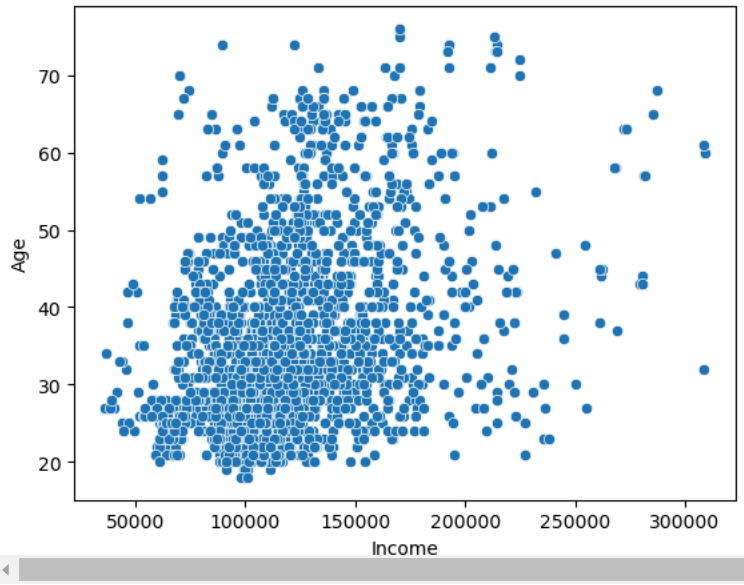

```
SS = data['Settlement size'].value_counts()
print(SS)
sns.countplot(x='Settlement size',data=data)
```

```
0    989
1    544
2    467
Name: Settlement size, dtype: int64
<Axes: xlabel='Settlement size', ylabel='count'>
```



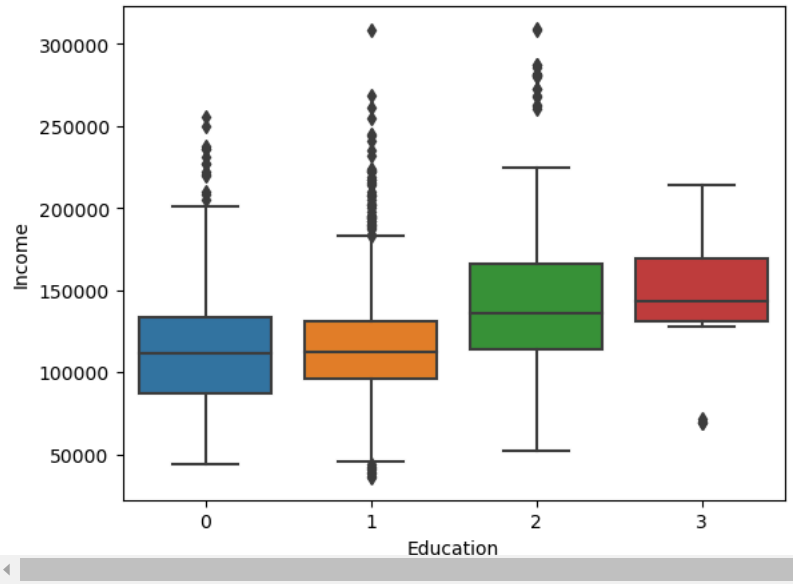

```
#lets see if the age is related to income
sns.scatterplot(x='Income',y='Age',data=data)

#age is not related to income
```

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



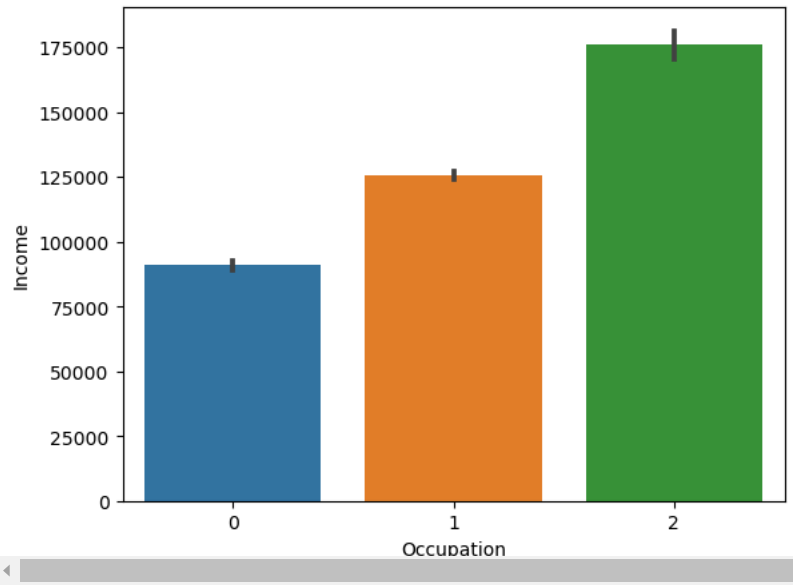

```
#lets see if the education is related to income
sns.boxplot(x='Education',y='Income',data=data)
#yes we can clearly see that income is related to education
```

```
<Axes: xlabel='Education', ylabel='Income'>
```



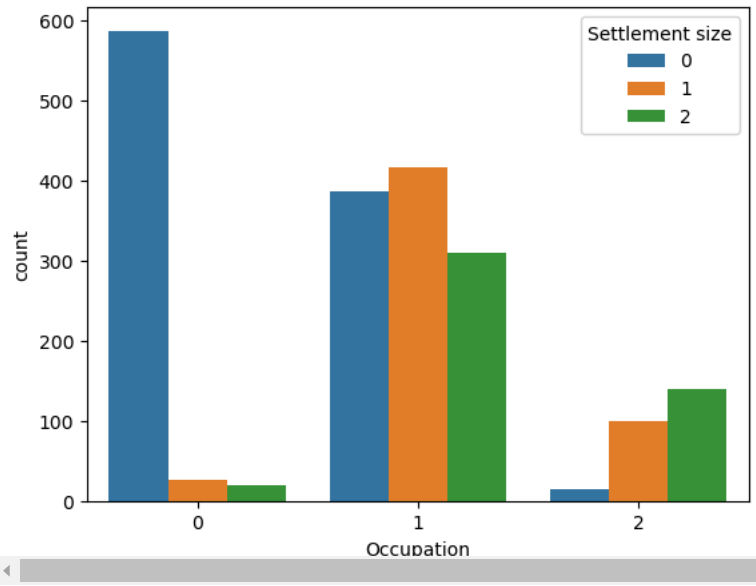

```
#lets see if the occupation is related to income
sns.barplot(x='Occupation',y='Income',data=data)
# so occupation is related to income
```

```
<Axes: xlabel='Occupation', ylabel='Income'>
```



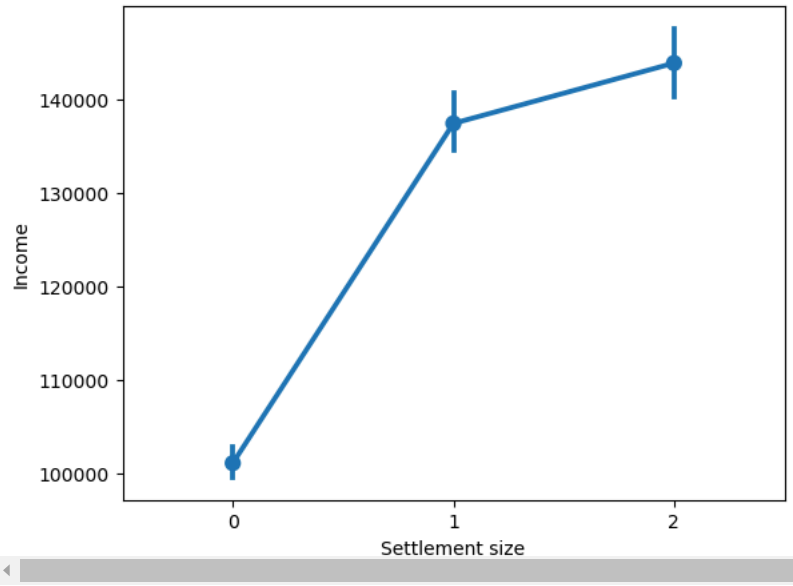

```
#lets see how income is related to settlement size
sns.countplot(x='Occupation',hue = 'Settlement size',data=data)
#we can say that occupation is related to settlement size
occ_ss = data.groupby('Occupation')['Settlement size'].value_counts()
print(occ_ss)
```

```
Occupation  Settlement size
0           0                587
            1                 27
            2                 19
1           1                417
            0                387
            2                309
2           2                139
            1                100
            0                 15
Name: Settlement size, dtype: int64
```




```
# lets see the relation between income and settlement size
sns.pointplot(x='Settlement size',y='Income',data=data)
#we can see the income is directly proportional to settlement size
```

```
<Axes: xlabel='Settlement size', ylabel='Income'>
```

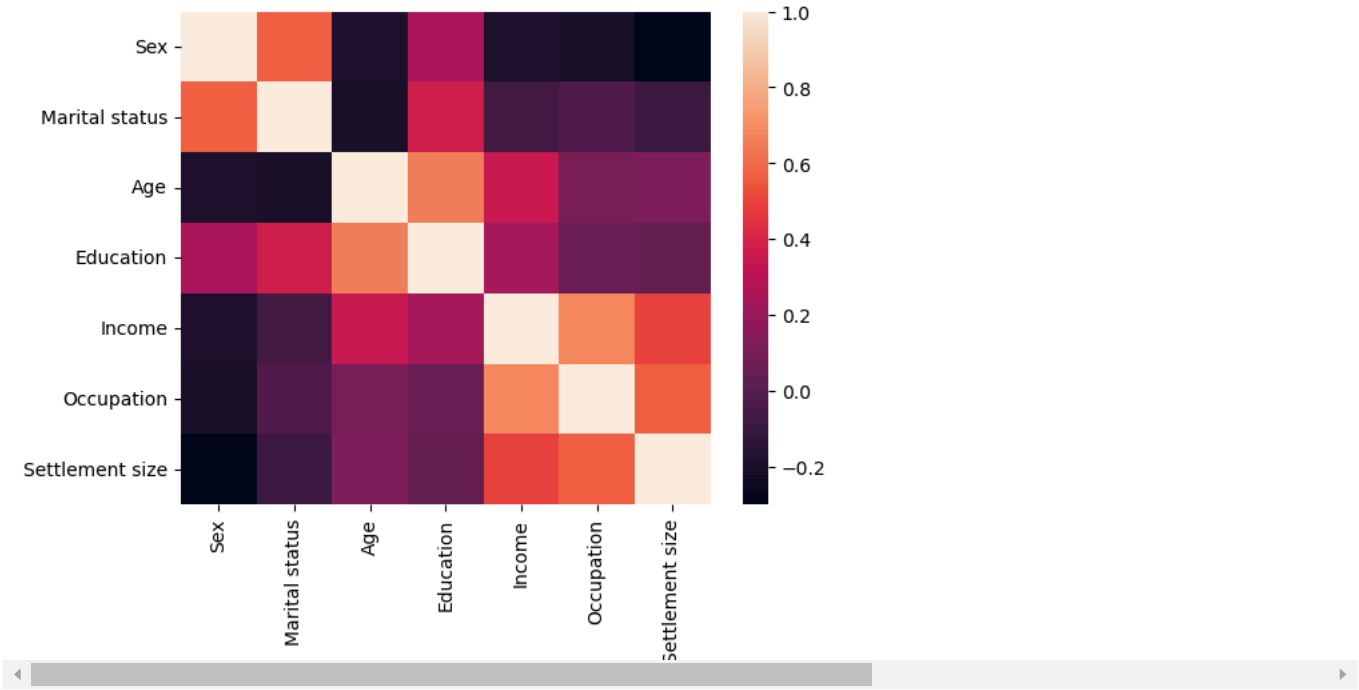



```
data.corr()
```

	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
Sex	1.000000	0.566511	-0.182885	0.244838	-0.195146	-0.202491	-0.300803
Marital status	0.566511	1.000000	-0.213178	0.374017	-0.073528	-0.029490	-0.097041
Age	-0.182885	-0.213178	1.000000	0.654605	0.340610	0.108388	0.119751
Education	0.244838	0.374017	0.654605	1.000000	0.233459	0.064524	0.034732
Income	-0.195146	-0.073528	0.340610	0.233459	1.000000	0.680357	0.490881
Occupation	-0.202491	-0.029490	0.108388	0.064524	0.680357	1.000000	0.571795
Settlement size	-0.300803	-0.097041	0.119751	0.034732	0.490881	0.571795	1.000000

```
sns.heatmap(data.corr())
```

```
<Axes: >
```



so we can say that the higher the education better the occupation and higher the income. Age , marital status and sex does not matter while clustering the data so we will go with education,occupation,income and settlement size

```
x= data[['Income','Settlement size','Education','Occupation']].values
```

```
from sklearn.cluster import KMeans
```

```
# deciding the number of clusters by using elbow method
wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
print(wcss)
```

```
[2903112757100.927, 1214579772524.6072, 678927294597.634, 377136830527.1168, 249071489411.37018, 177064947566.27502, 129716147560.46
```

plotting the elbow method

```
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='-', color='b')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
new_ticks = [1, 2, 3, 4, 5, 6,7,8,9,10]
plt.xticks(new_ticks)

plt.show()
```