In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\10_USA_Housing.csv")
df
```

Out[2]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	A
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael F 674\nLaural
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnsc Suite 07 Kathle
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 E Stravenue\nDar WI
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\n
4							USNS Raymor

In [3]:
```
df.head()
```

Out[3]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addres
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Ap 674\nLaurabury, N 3701
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson View Suite 079\nLak Kathleen, CA
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabe Stravenue\nDanieltow WI 06482
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO A 4482
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFP AE 0938

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

In [6]:

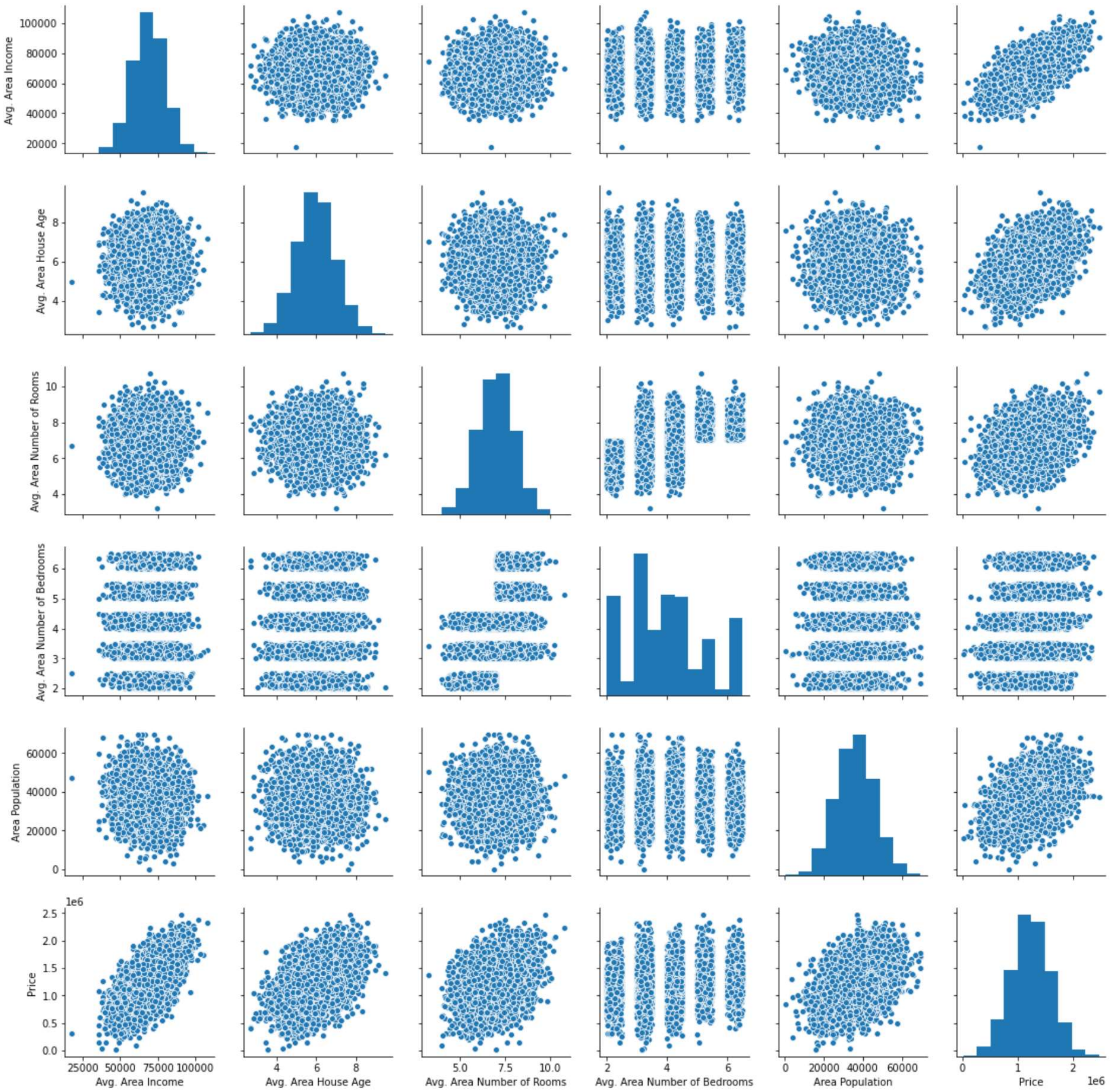
```
df.columns
```

Out[6]:

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Room
s',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Addres
s'],
      dtype='object')
```

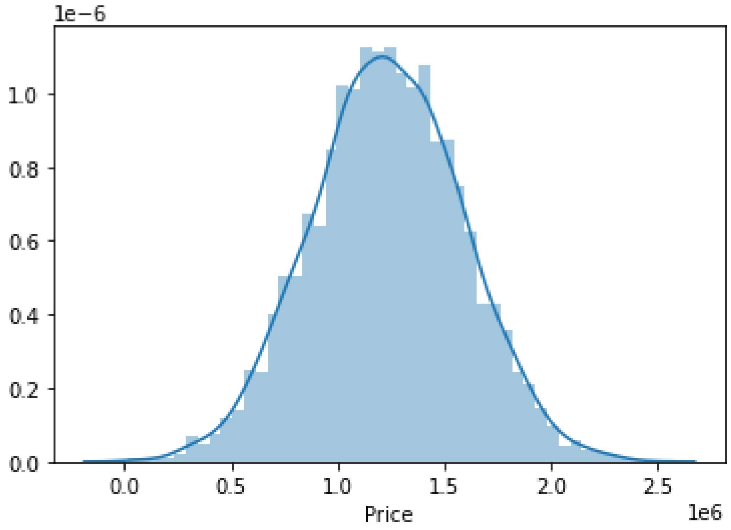
In [7]:
```
sns.pairplot(df)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x2bde6e21a30>
```



In [8]:
```
sns.distplot(df['Price'])
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2bde7cae610>
```



In [9]:
```
df1=df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price']]
df1
```

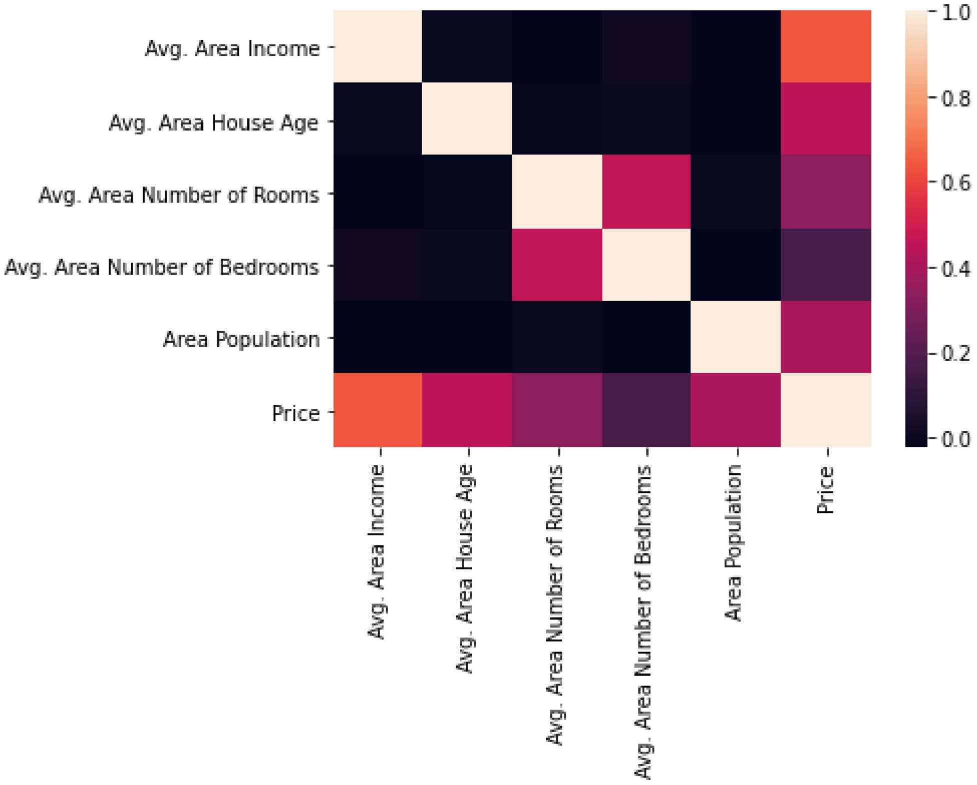
Out[9]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05
...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06

5000 rows × 6 columns

In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2bde84f6ac0>
```



In [11]:
```
x=df1[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]
y=df1['Price']
```

In [12]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [13]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
print(lr.intercept_)
```

```
-2638957.114938198
```

In [15]:

```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[15]:

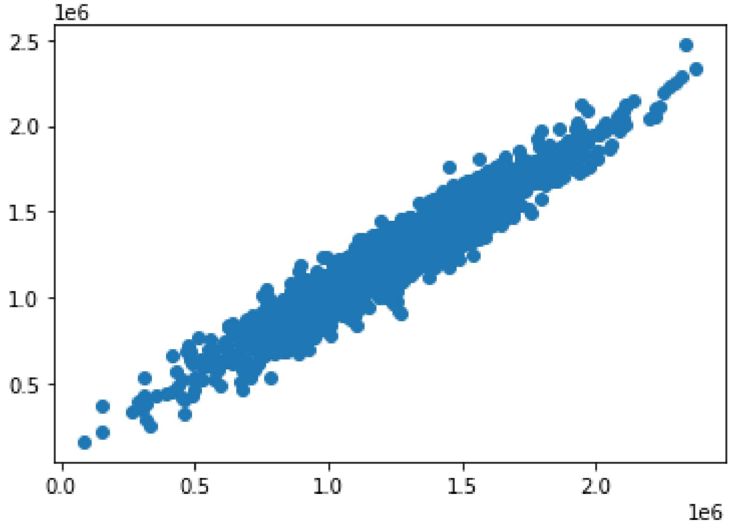
	Co-efficient
Avg. Area Income	21.531717
Avg. Area House Age	166600.643364
Avg. Area Number of Rooms	119726.455133
Avg. Area Number of Bedrooms	2562.636684
Area Population	15.244846

In [16]:

```
prediction= lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x2bde9f97eb0>
```



In [17]:

```
print(lr.score(x_test,y_test))
```

```
0.9164804208931521
```

In [18]:

```
print(lr.score(x_train,y_train))
```

```
0.9186122205871261
```

In [19]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]:

```
Ridge(alpha=10)
```

In [21]:

```
rr.score(x_test,y_test)
```

Out[21]:

```
0.9164860867254999
```

```
In [22]: la=Lasso(alpha=10)
         la.fit(x_train,y_train)

Out[22]: Lasso(alpha=10)
```

```
In [23]: la.score(x_test,y_test)

Out[23]: 0.9164809475800579
```

```
In [26]: from sklearn.linear_model import ElasticNet
         en=ElasticNet()
         en.fit(x_train,y_train)

Out[26]: ElasticNet()
```

```
In [27]: print(en.intercept_)

         -2040587.6936731543
```

```
In [28]: print(en.predict(x_test))

         [1084249.01236266 1044344.4566849  1342069.10473779 ... 1195992.20497631
           402744.02547299 1056667.69170541]
```

```
In [29]: print(en.score(x_test,y_test))

         0.8829596809738758
```

Evaluation metrics

```
In [31]: from sklearn import metrics
```

```
In [32]: print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))

         Mean Absolute Error 81956.46667199011
```

```
In [35]: print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))

         Mean Squared Error 10388412415.294918
```

```
In [36]: print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prec

         Root Mean Squared Error: 101923.56162975721
```

```
In [ ]: 
```