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\user\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	
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael 674\nLaur
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 John Suite (Kathl
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Stravenue\nD W
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raym
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Willia AP 30
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 8489\nAPO A
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Trac Suite 076\nJo
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 Geor Apt. 509\nE

5000 rows × 7 columns

In [3]:

```
df.head(10)
```

Out[3]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Ad
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Fer 674\nLaurabu 3
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Suite 079\ Kathleen
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Eliz Stravenue\nDanie WI 06
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFF
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\ AE
5	80175.754159	4.988408	6.104512	4.04	26748.428425	1.068138e+06	06039 Jennifer Is Apt. 443\nTrac
6	64698.463428	6.025336	8.147760	3.41	60828.249085	1.502056e+06	4759 Daniel S 442\nNguyenburg
7	78394.339278	6.989780	6.620478	2.42	36516.358972	1.573937e+06	972 Viaduct\nLake W TN 17778
8	59927.660813	5.362126	6.393121	2.30	29387.396003	7.988695e+05	USS Gilbert\nFF
9	81885.927184	4.423672	8.167688	6.10	40149.965749	1.545155e+06	Unit 944 0958\nDPO AE

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]:

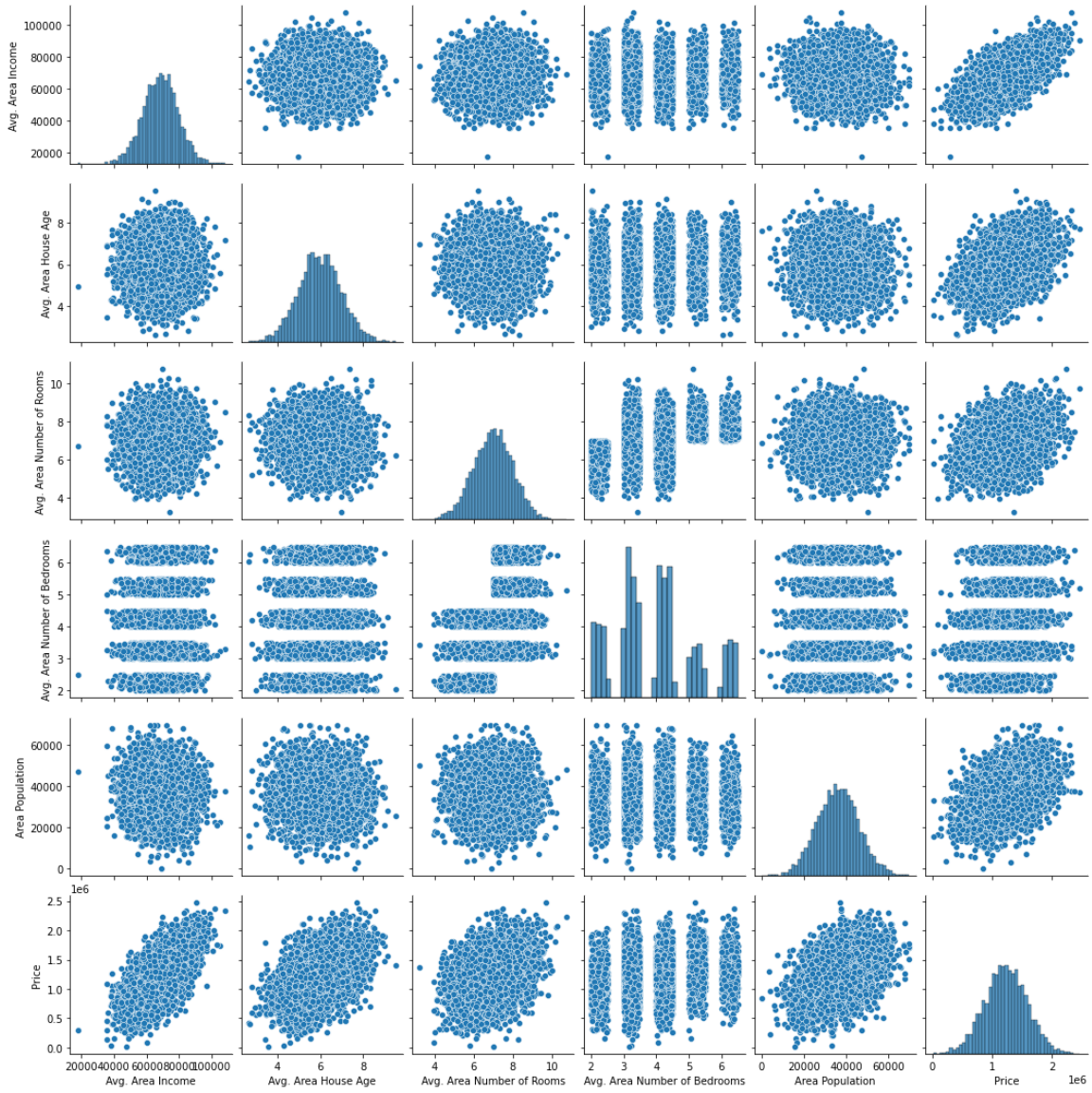
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Roo
ms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Addres
s'],
      dtype='object')
```

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1c8d5361b50>
```



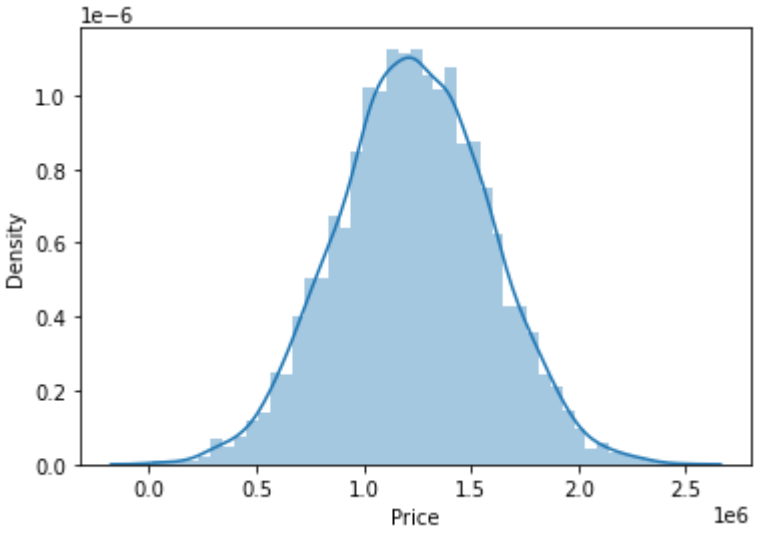
In [8]:

```
sns.distplot(df['Price'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:

```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```

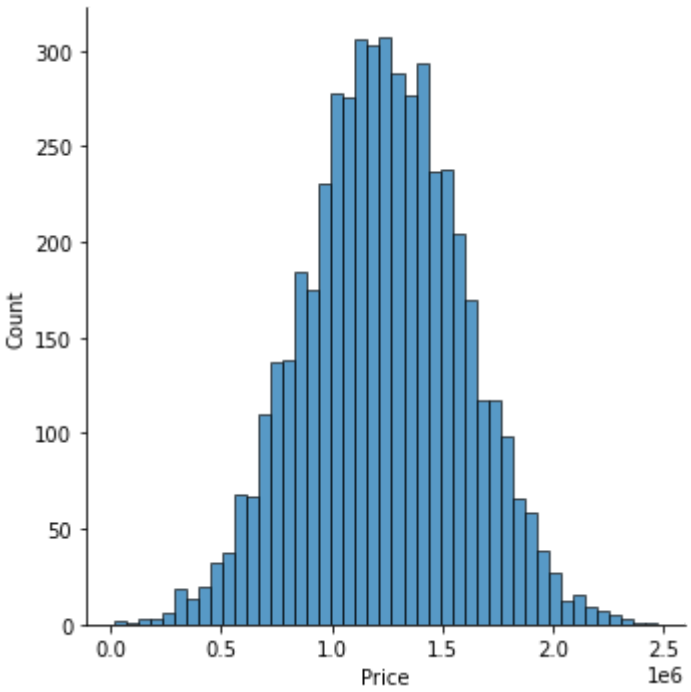


In [9]:

```
sns.displot(df["Price"])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1c8d05c0310>
```



In [10]:

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

In [11]:

```
sns.heatmap(df1.corr())
```

Out[11]:

<AxesSubplot:>

In [12]:

```
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 [13]:

```
from sklearn.model_selection import train_test_split
```

In [14]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [15]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for
```

Out[15]:

```
LinearRegression()
```

In [16]:

```
print(lr.intercept_)
```

```
[-2645390.33723764]
```

In [17]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

Out[17]:

	0	1	2	3	4
0	21.589287	165678.030232	120943.020651	3485.792051	15.104428

In [18]:

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

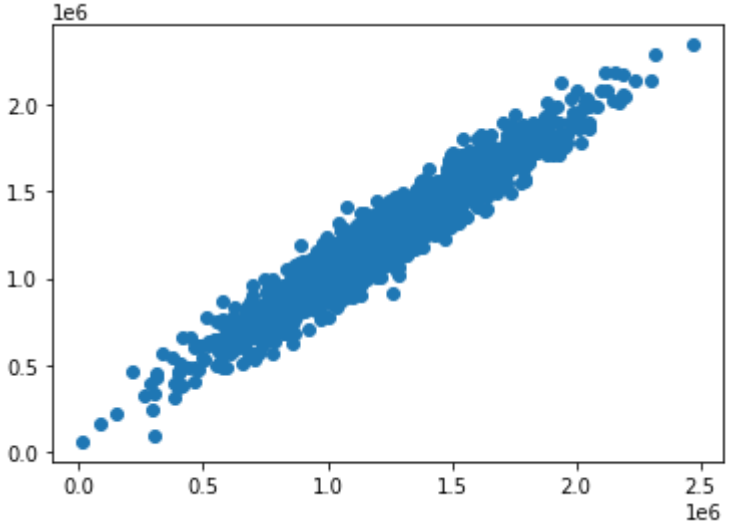
```
0.9227842269582164
```

In [19]:

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

Out[19]:

```
<matplotlib.collections.PathCollection at 0x1c8d8db15b0>
```



In [20]:

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

Out[20]:

```
0.9227842269582164
```

In [21]:

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

Out[21]:

```
0.9157829496752972
```

In [22]:

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

In [23]:

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

Out[23]:

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

In [24]:

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

Out[24]:

```
0.922787550110281
```

In [25]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[25]:

```
Lasso(alpha=10)
```

In [26]:

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

Out[26]:

```
0.9227853233896435
```

Elastic Net

In [27]:

```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[27]:

```
ElasticNet()
```

In [28]:

```
print(en.coef_)
```

```
[2.15091862e+01 1.08234702e+05 7.55105892e+04 1.58272860e+04
 1.49415272e+01]
```

In [29]:

```
print(en.intercept_)
```

```
[-2022298.62225742]
```

In [30]:

```
prediction=en.predict(x_test)
print(prediction)
```

```
[1442502.90281944 1102186.10441095 1466488.65418177 ... 1656575.04713553
 1551865.68698104 1179855.46817265]
```

In [31]:

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

```
0.8840236634865118
```

Evaluation Metrics

In [32]:

```
from sklearn import metrics
```

In [33]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 98458.85783150232
```

In [34]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 14935741954.46375
```

In [35]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 122211.87321395475
```