In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [30]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\10_USA_Housing.csv")
df
```

Out[30]:

	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 Ferr 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
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\ AP 30153
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258 8489\nAPO AA 4
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy G Suite 076\nJoshu VA
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFF
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George F Apt. 509\nEast N

5000 rows × 7 columns

In [31]:
```
df.head(10)
```

Out[31]:

	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 Viev Suite 079\nLal 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 448
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFP AE 093
5	80175.754159	4.988408	6.104512	4.04	26748.428425	1.068138e+06	06039 Jennifer Islan Apt. 443\nTracypo KS
6	64698.463428	6.025336	8.147760	3.41	60828.249085	1.502056e+06	4759 Daniel Shoa Sui 442\nNguyenburgh, C
7	78394.339278	6.989780	6.620478	2.42	36516.358972	1.573937e+06	972 Joy Viaduct\nLake Willia TN 17778-64
8	59927.660813	5.362126	6.393121	2.30	29387.396003	7.988695e+05	USS Gilbert\nFPO A 209
9	81885.927184	4.423672	8.167688	6.10	40149.965749	1.545155e+06	Unit 9446 B 0958\nDPO AE 970

In [32]:
```
df.isnull().sum()
```

Out[32]:
```
Avg. Area Income              0
Avg. Area House Age           0
Avg. Area Number of Rooms     0
Avg. Area Number of Bedrooms  0
Area Population               0
Price                         0
Address                       0
dtype: int64
```

In [34]:
```
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 [36]:
```
df.describe()
```

Out[36]:

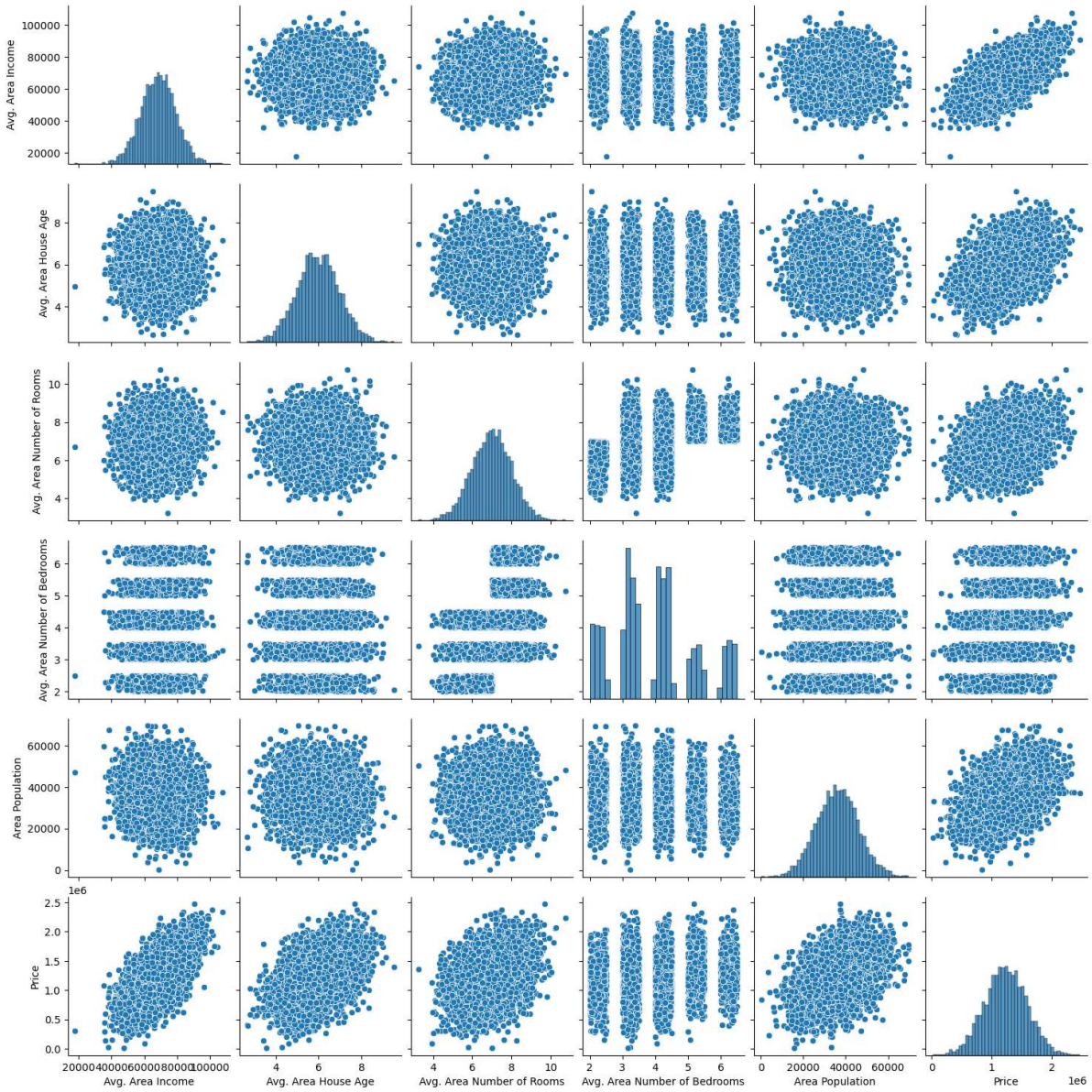
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	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 [38]:
```
df.columns
```

Out[38]:
```
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 [39]:
```
sns.pairplot(df)
```

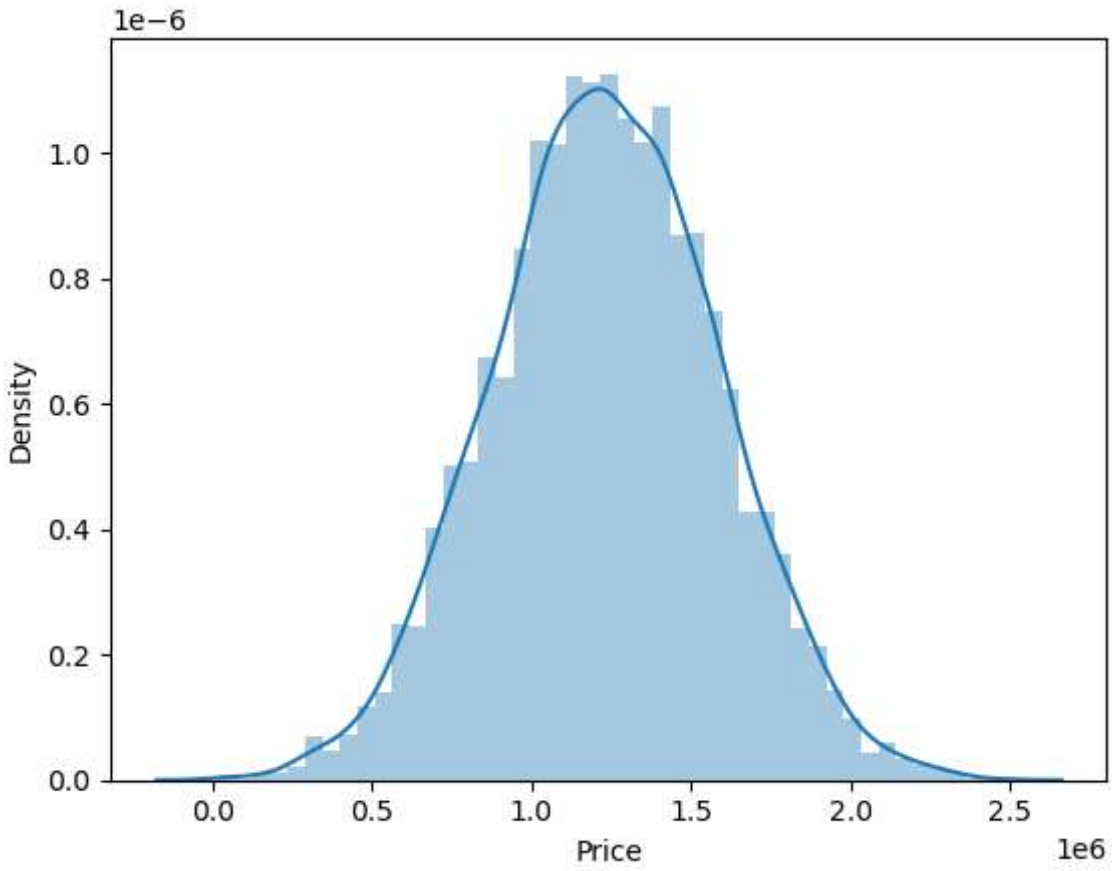
Out[39]:
```
<seaborn.axisgrid.PairGrid at 0x1a4e4fefb10>
```



In [41]:
```
sns.distplot(df["Price"])
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_11940\941010651.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

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

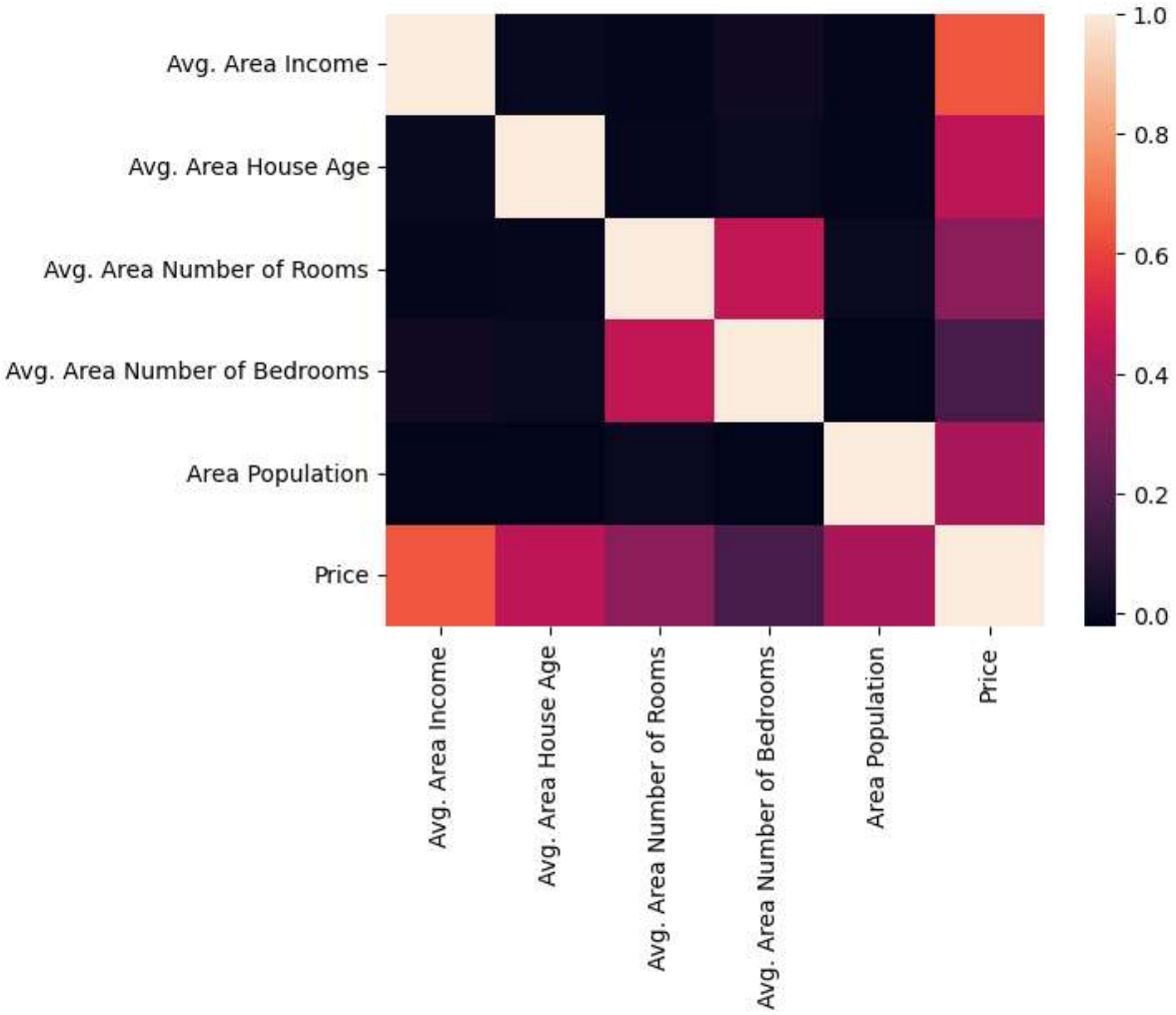
Out[41]: <Axes: xlabel='Price', ylabel='Density'>



In [43]:
```
df1=df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms'
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price']]
```

In [44]:
```
sns.heatmap(df1.corr())
```

Out[44]: <Axes: >



In [54]:
```
df1.drop_duplicates(inplace=True)
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_11940\4156330626.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df1.drop_duplicates(inplace=True)
```

In [55]:
```
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 [56]:
```
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 [57]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[57]:
```
LinearRegression
LinearRegression()
```

In [58]:
```
print(lr.intercept_)
```

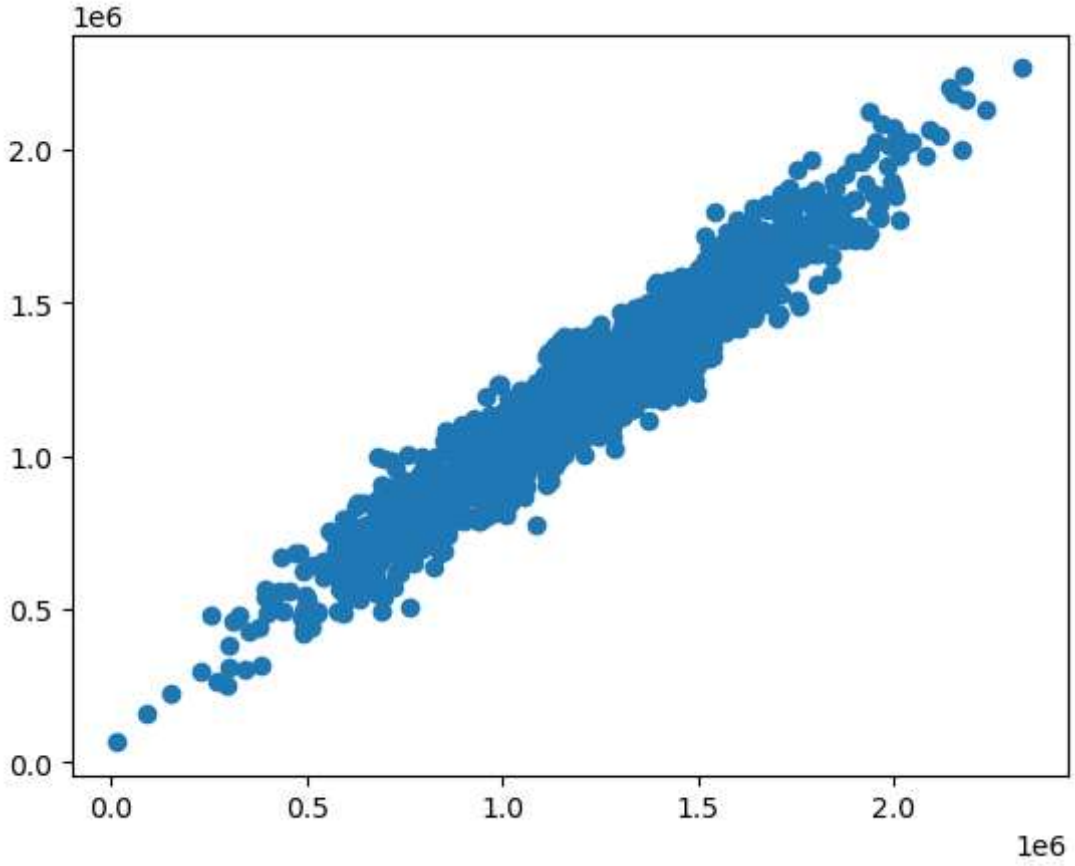
```
[-2632882.61590988]
```

In [50]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[50], line 1
----> 1 coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
      2 coeff

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:722, in DataFrame.__i
nit__(self, data, index, columns, dtype, copy)
    712         mgr = dict_to_mgr(
    713             # error: Item "ndarray" of "Union[ndarray, Series, Inde
x]" has no
    714             # attribute "name"
   (...)
    719             typ=manager,
    720         )
    721     else:
--> 722         mgr = ndarray_to_mgr(
    723             data,
    724             index,
    725             columns,
    726             dtype=dtype,
    727             copy=copy,
    728             typ=manager,
    729         )
    731 # For data is list-like, or Iterable (will consume into list)
    732 elif is_list_like(data):

File ~\anaconda3\Lib\site-packages\pandas\core\internals\construction.py:349,
in ndarray_to_mgr(values, index, columns, dtype, copy, typ)
    344 # _prep_ndarraylike ensures that values.ndim == 2 at this point
    345 index, columns = _get_axes(
    346     values.shape[0], values.shape[1], index=index, columns=columns
    347 )
--> 349 _check_values_indices_shape_match(values, index, columns)
    351 if typ == "array":
    353     if issubclass(values.dtype.type, str):

File ~\anaconda3\Lib\site-packages\pandas\core\internals\construction.py:420,
in _check_values_indices_shape_match(values, index, columns)
    418 passed = values.shape
    419 implied = (len(index), len(columns))
--> 420 raise ValueError(f"Shape of passed values is {passed}, indices imply
{implied}")

ValueError: Shape of passed values is (1, 5), indices imply (5, 1)
```

In [59]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[59]: <matplotlib.collections.PathCollection at 0x1a4f2ef7c90>



In [60]:
```
print(lr.score(x_test,y_test))
```

```
0.9219418105240463
```

In [ ]: