problem statement

A real estate agent want help to predict the house price for region in USA. He gave us the dataset to work on to us Linear Regression model. Create a model that help him to estiamte of what the house would sell for.

Data Collection

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	Ad
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferr 674\nLaurabur 3
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Suite 079\n Kathleen,
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Eliz Stravenue\nDaniel WI 06
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFP 4
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\n AE
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\n 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\nJoshua VA
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFP 7
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George R Apt. 509\nEast N

5000 rows × 7 columns

In [3]:
```
# to display info
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 [4]:
```
# t display summerize the data
df.describe()
```

Out[4]:

	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

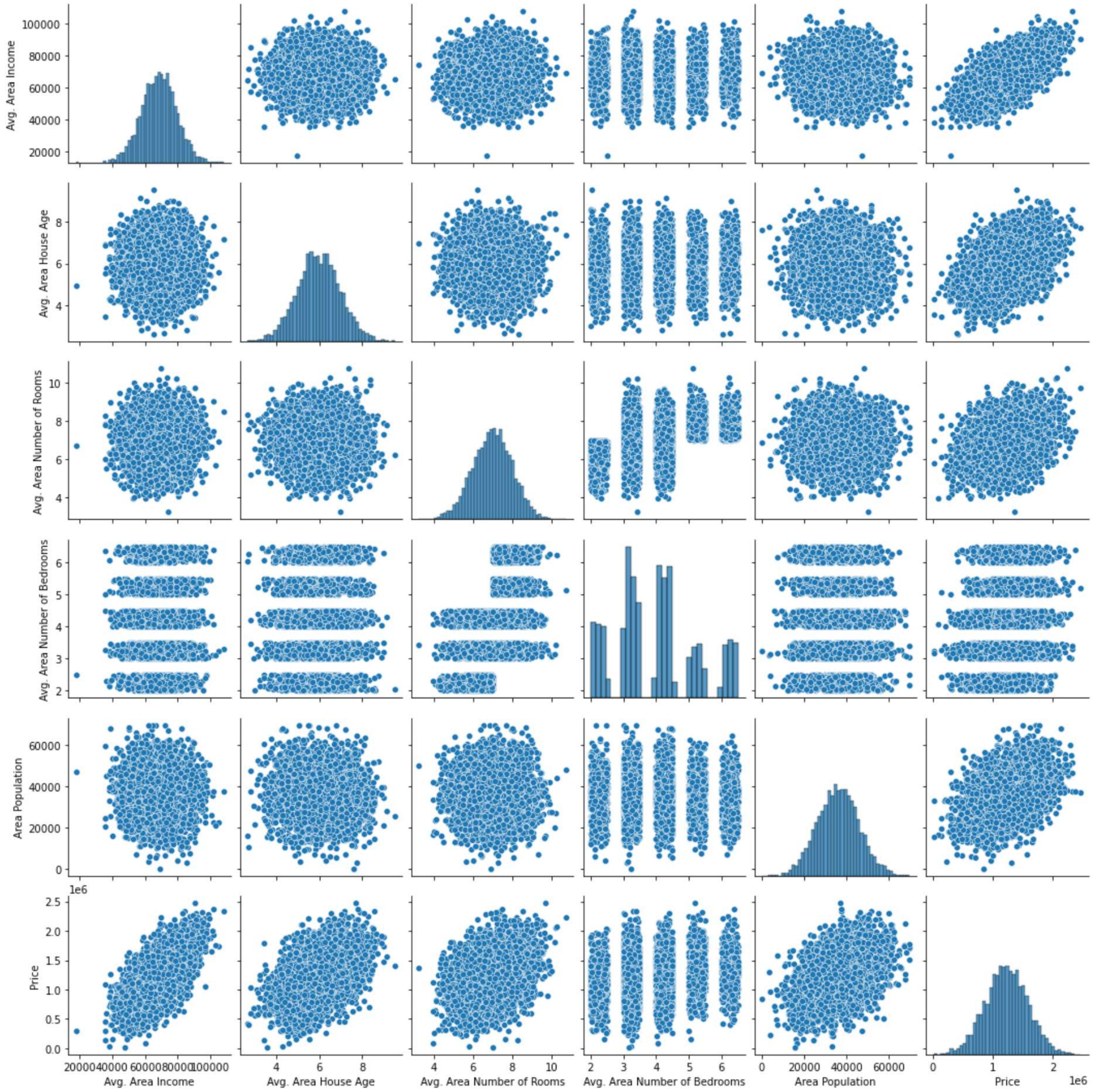
In [5]:
```
# to display columes
df.columns
```

Out[5]:
```
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')
```

EDA and visualization

In [6]:
```
sns.pairplot(df)
```

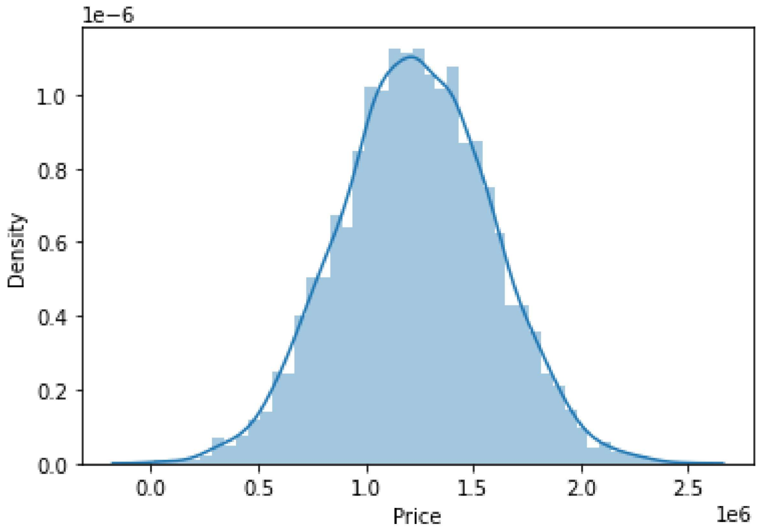
Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x2b7169de9a0>
```



In [7]:
```
# to display distribution graph for price column
sns.distplot(df['Price'])
```

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

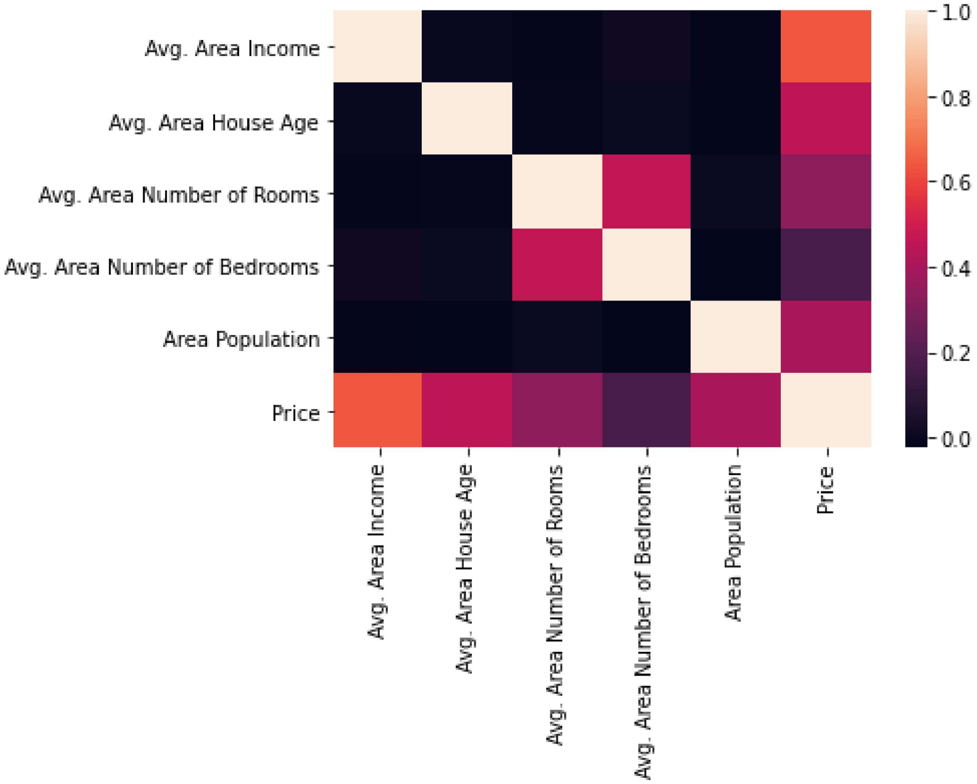
Out[7]:
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



In [8]:
```
df1 = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price']]
```

In [9]:
```
# correlation map to find relationship
sns.heatmap(df1.corr())
```

Out[9]:
```
<AxesSubplot:>
```



To Trait the model - model building

we are going to train linear regression model; we are going to split data into two variable x and y where x is independent variable(input) and y is dependent on x (output) we could ignore address column as it in not required for our model

In [10]:
```
# Assign x and y for linear regression
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 [11]:
```
# to split dataset into training data and test data

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 [12]:

```
#Linear Regression

from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x_train,y_train)
```

Out[12]:

```
LinearRegression()
```

In [13]:

```
# intercept is value of c
print(lr.intercept_)
```

```
-2623528.589101932
```

In [14]:

```
# co-efficient value of m
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[14]:

	Co-efficient
Avg. Area Income	21.556938
Avg. Area House Age	165030.583398
Avg. Area Number of Rooms	119820.614710
Avg. Area Number of Bedrooms	148.133484
Area Population	15.275916

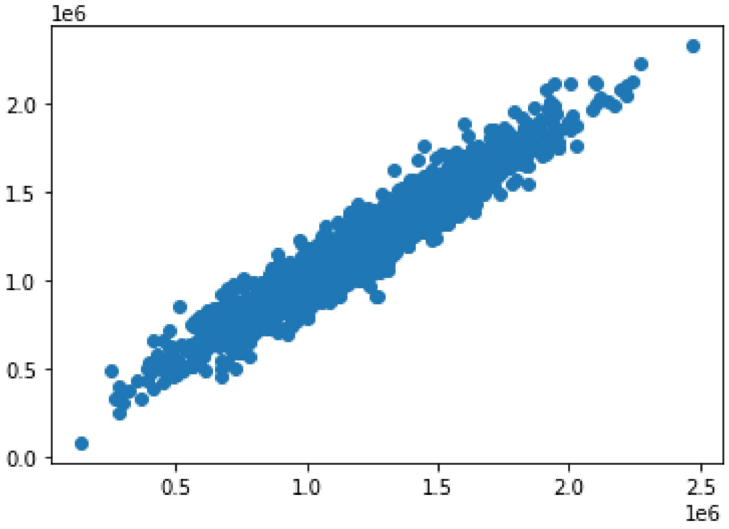
In [15]:

```
#predict the graph in linear regression graph

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

Out[15]:

```
<matplotlib.collections.PathCollection at 0x2b71b30cbe0>
```



```
In [16]: #Accuracy of linear regression

         print(lr.score(x_test,y_test))

         0.9169489109126523
```

```
In [17]: lr.score(x_train,y_train)

Out[17]: 0.9183894543649431
```

```
In [18]: from sklearn.linear_model import Ridge,Lasso
```

```
In [20]: rr = Ridge(alpha=10)
         rr.fit(x_train,y_train)
         rr.score(x_test,y_test)

Out[20]: 0.918386485414195
```

```
In [21]: rr.score(x_train,y_train)

Out[21]: 0.918386485414195
```

```
In [24]: lr = Lasso(alpha=10)
         lr.fit(x_train,y_train)
         lr.score(x_test,y_test)

Out[24]: 0.916947302511081
```

```
In [25]: lr.score(x_train,y_train)

Out[25]: 0.9183894526053276
```

```
In [ ]:
```