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

%matplotlib inline
```

In [2]:
```
df=pd.read_csv("C:/Users/hp/Downloads/USA_Housing.csv")
```

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	Addre
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry A 674\nLaurabury, N 370
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Vie Suite 079\nLa Kathleen, CA
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabe Stravenue\nDanieltov 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\nFF AE 093

In [4]:
```
df.tail()
```

Out[4]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

In [5]:
```
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
```

we have 7 columns and 5000 rows

In [6]:
```
df.describe()
```

Out[6]:

	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 [8]:
```
df.columns
```

Out[8]:
```
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')
```

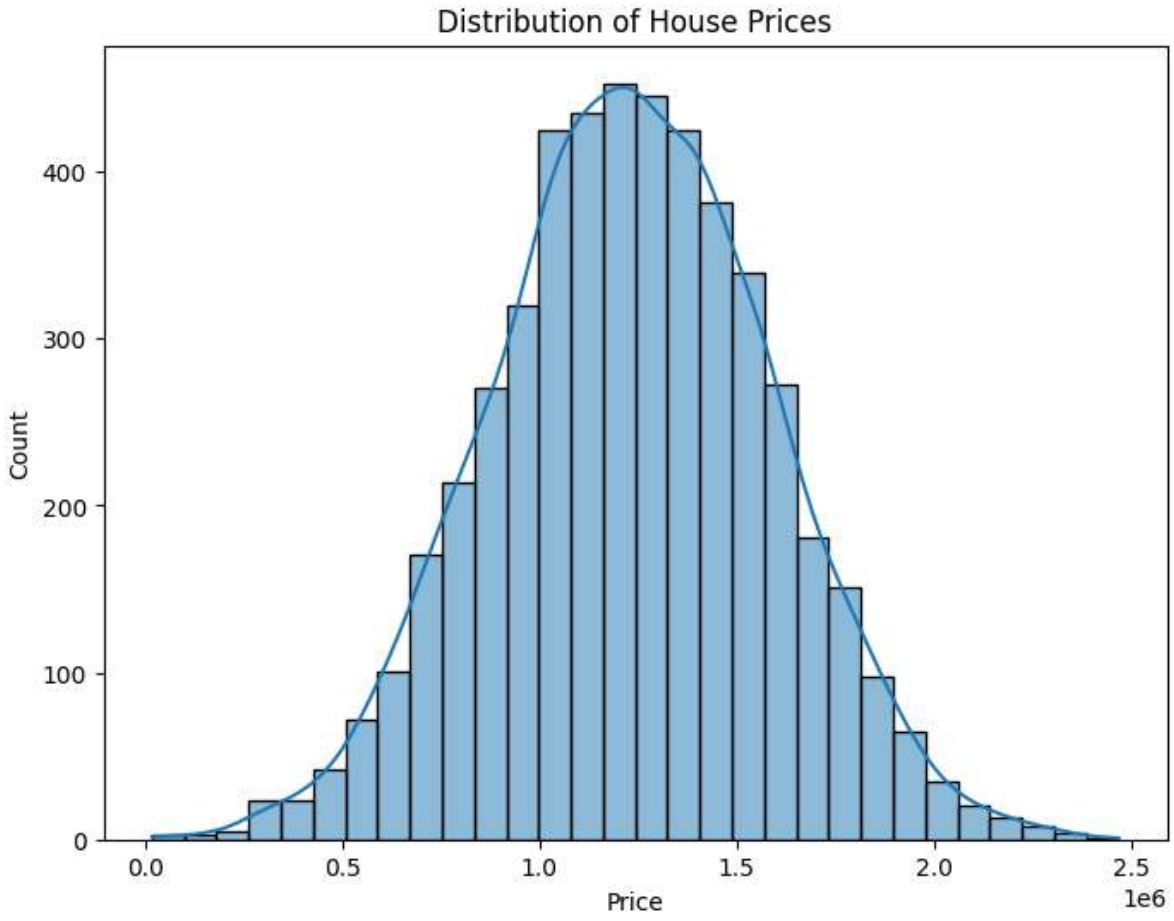
EXPLORATORY DATA ANALYSIS

In [10]:
```
print(df.isnull().sum())
```

```
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
```

There is no null values present in the dataset.

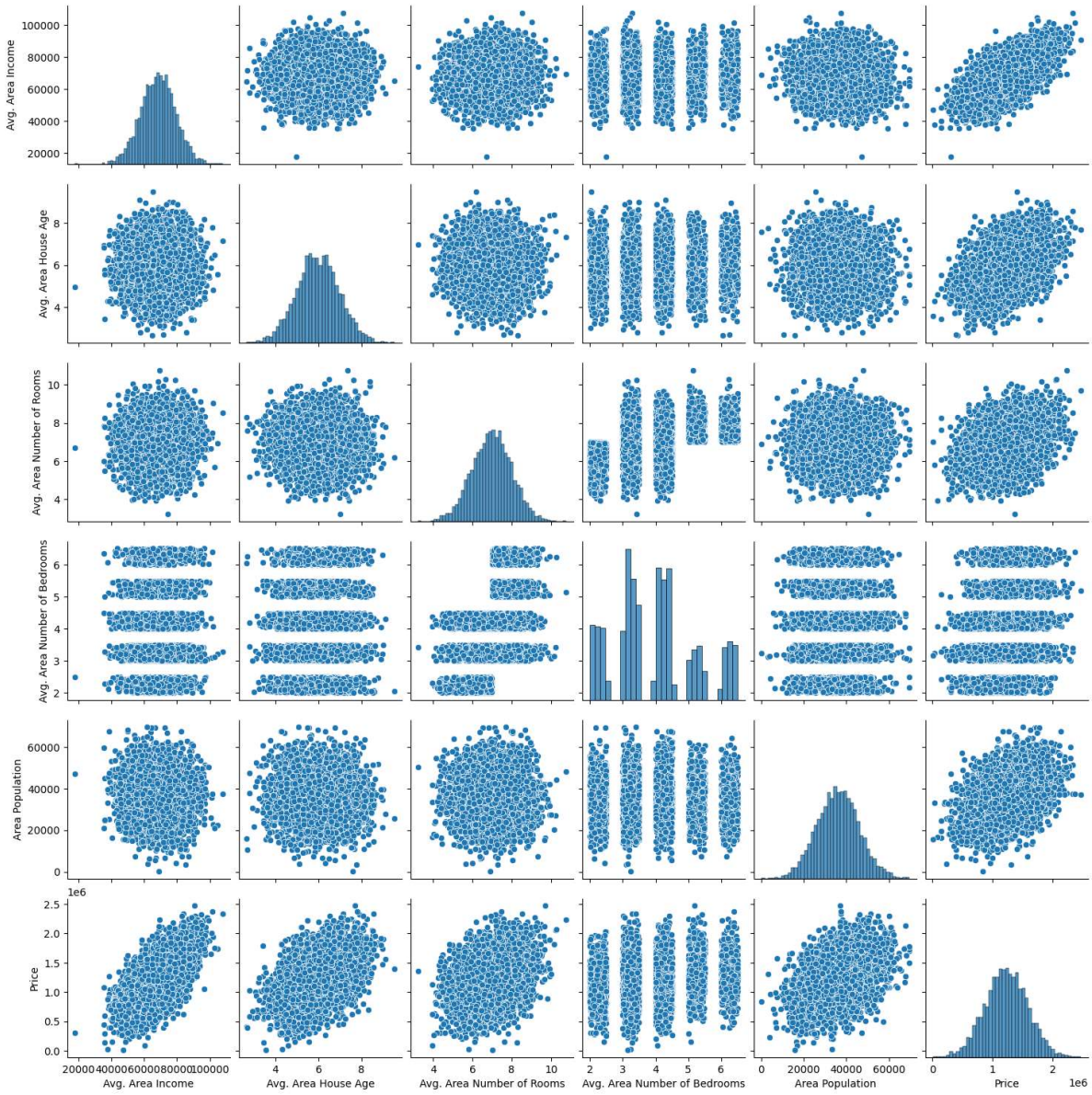
In [14]:
```
plt.figure(figsize=(8,6))
sns.histplot(df['Price'],bins=30,kde=True)
plt.title('Distribution of House Prices')
plt.xlabel('Price')
plt.ylabel('Count')
plt.show()
```



We see the distribution of target variables Skewness and spread, so that it helps us to select appropriate statistical data models etc.

In [17]:
```
plt.figure(figsize=(12,8))
sns.pairplot(df[['Avg. Area Income','Avg. Area House Age', 'Avg. Area Number o
       'Avg. Area Number of Bedrooms', 'Area Population','Price']])
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



we see the relationship between the target variables and other features

In [18]:
```
sns.heatmap(df.corr(), annot=True)
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_21272\621126171.py:1: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a futur
e version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot=True)
```

Out[18]: <AxesSubplot: >

Correlation matrix between features. The relation between householder's income and house price is high

MODEL BUILDING

In [20]:
```
X = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms'
                'Avg. Area Number of Bedrooms', 'Area Population']]

y = df['Price']
```

In [21]:
```
#Split data into trainig and testing

from sklearn.model_selection import train_test_split
```

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, rando
```

In [23]:
```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train,y_train)
```

Out[23]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [26]:
```
# make prediction using test data
y_pred = lm.predict(X_test)
```

In [28]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

In [29]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

In [30]:
```
print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"R-squared (R²): {r2}")
```

```
Mean Squared Error (MSE): 10460958907.208803
Mean Absolute Error (MAE): 82288.22251914945
R-squared (R²): 0.9176824009649256
```

Y is 90% dependent on x variables

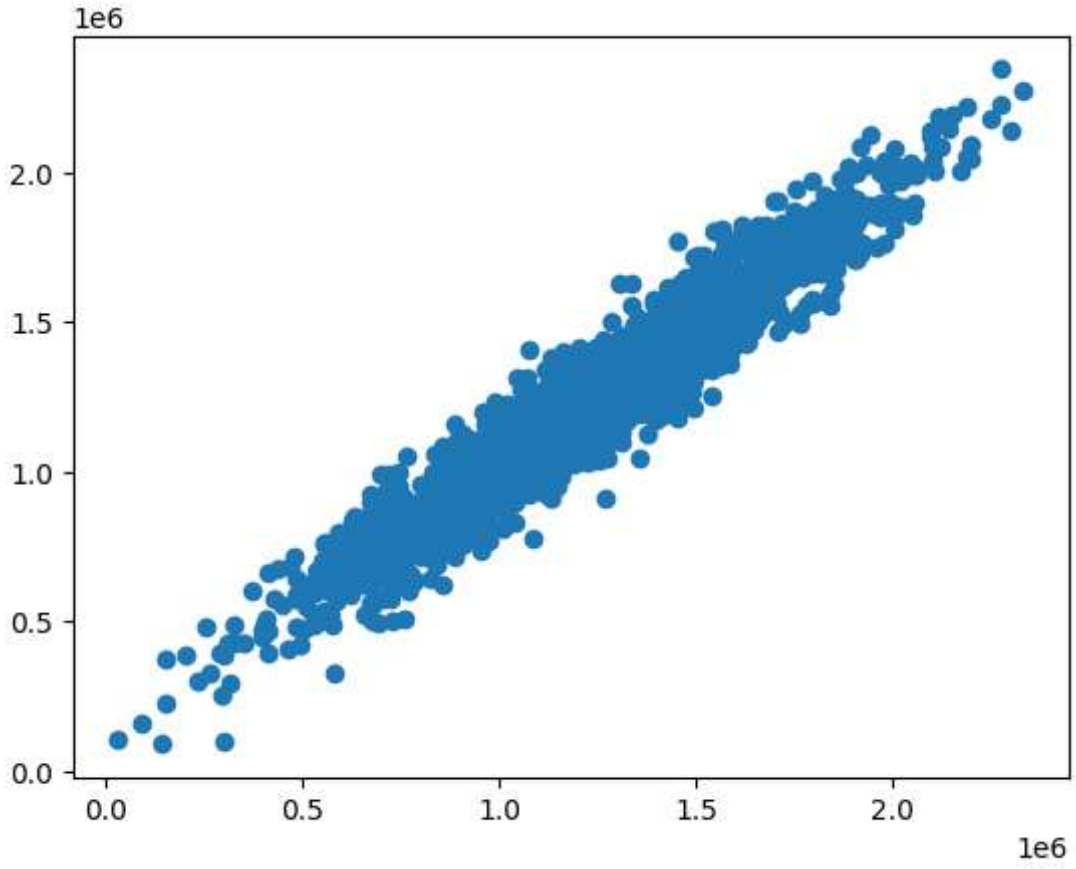
In [33]:
```
# Interpret the coefficients
coefficients = pd.DataFrame({'feature': X.columns, 'coefficient': lm.coef_})
print(coefficients)
```

```
                          feature    coefficient
0              Avg. Area Income      21.528276
1           Avg. Area House Age  164883.282027
2     Avg. Area Number of Rooms  122368.678027
3  Avg. Area Number of Bedrooms    2233.801864
4               Area Population      15.150420
```

In [36]:
```
predictions = lm.predict(X_test)
```

In [37]:
```
plt.scatter(y_test,predictions)
```

Out[37]:
```
<matplotlib.collections.PathCollection at 0x20312cd3c10>
```



Here we got a linear scatter plot. It shows that model is fit and good.

In [ ]:
```

```