In [21]:

```
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [6]:

```
df=pd.read_csv(r"C:\Users\91949\Downloads\USA_Housing.csv")
df
```

Out[6]:

	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 Michae 674\nLa
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Joh Suite Katl
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	912 Stravenue\nl
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barne
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Rayr
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Wil AP 3
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 8489\nAPC
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tr Suite 076\n
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallac
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 Gec Apt. 509\

5000 rows × 7 columns

In [7]:

```
df.head()
```

Out[7]:

	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 Fe 674\nLaurab
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnso Suite 07 Kathlee
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 E Stravenue\nDan WI (
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nl
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymon AE

In [8]:

```
df.tail()
```

Out[8]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Ad
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	U Williams\r AP 30153-
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258 8489\nAP 42991-
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Garden 076\nJoshua VA
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	Wallace\r AE 7
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 Ge Ridge 509\nEast N

In [9]:

```
df.describe()
```

Out[9]:

	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 [10]:

```
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 [12]:

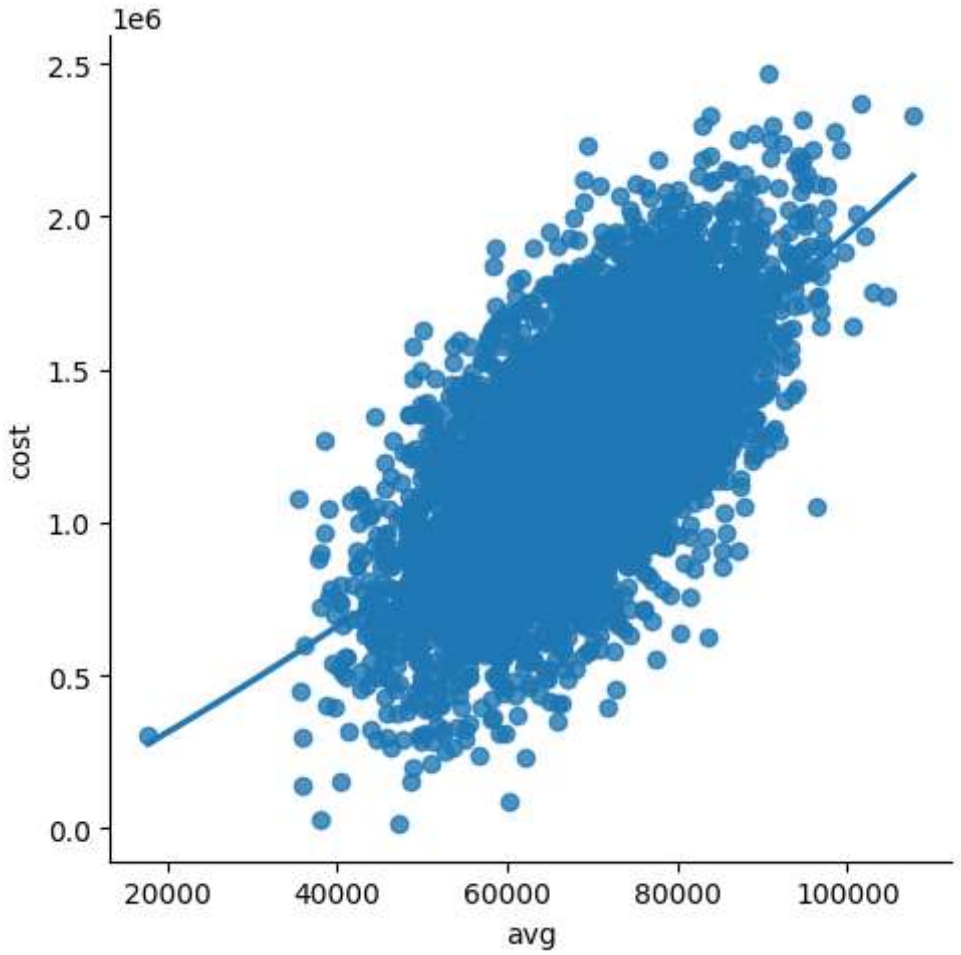
```
df=df[['Avg. Area Income','Price']]
df.columns=['avg','cost']
```

In [13]:

```
sns.lmplot(x='avg',y='cost',data=df,order=2,ci=None)
```

Out[13]:

```
<seaborn.axisgrid.FacetGrid at 0x215c36b7250>
```



In [14]:

```
df.fillna(method='ffill')
```

Out[14]:

	avg	cost
0	79545.458574	1.059034e+06
1	79248.642455	1.505891e+06
2	61287.067179	1.058988e+06
3	63345.240046	1.260617e+06
4	59982.197226	6.309435e+05
...	...	...
4995	60567.944140	1.060194e+06
4996	78491.275435	1.482618e+06
4997	63390.686886	1.030730e+06
4998	68001.331235	1.198657e+06
4999	65510.581804	1.298950e+06

5000 rows × 2 columns

In [15]:

```
df.dropna()
```

Out[15]:

	avg	cost
0	79545.458574	1.059034e+06
1	79248.642455	1.505891e+06
2	61287.067179	1.058988e+06
3	63345.240046	1.260617e+06
4	59982.197226	6.309435e+05
...	...	...
4995	60567.944140	1.060194e+06
4996	78491.275435	1.482618e+06
4997	63390.686886	1.030730e+06
4998	68001.331235	1.198657e+06
4999	65510.581804	1.298950e+06

5000 rows × 2 columns

In [19]:

```
x=np.array(df['avg']).reshape(-1,1)
y=np.array(df['cost']).reshape(-1,1)
```

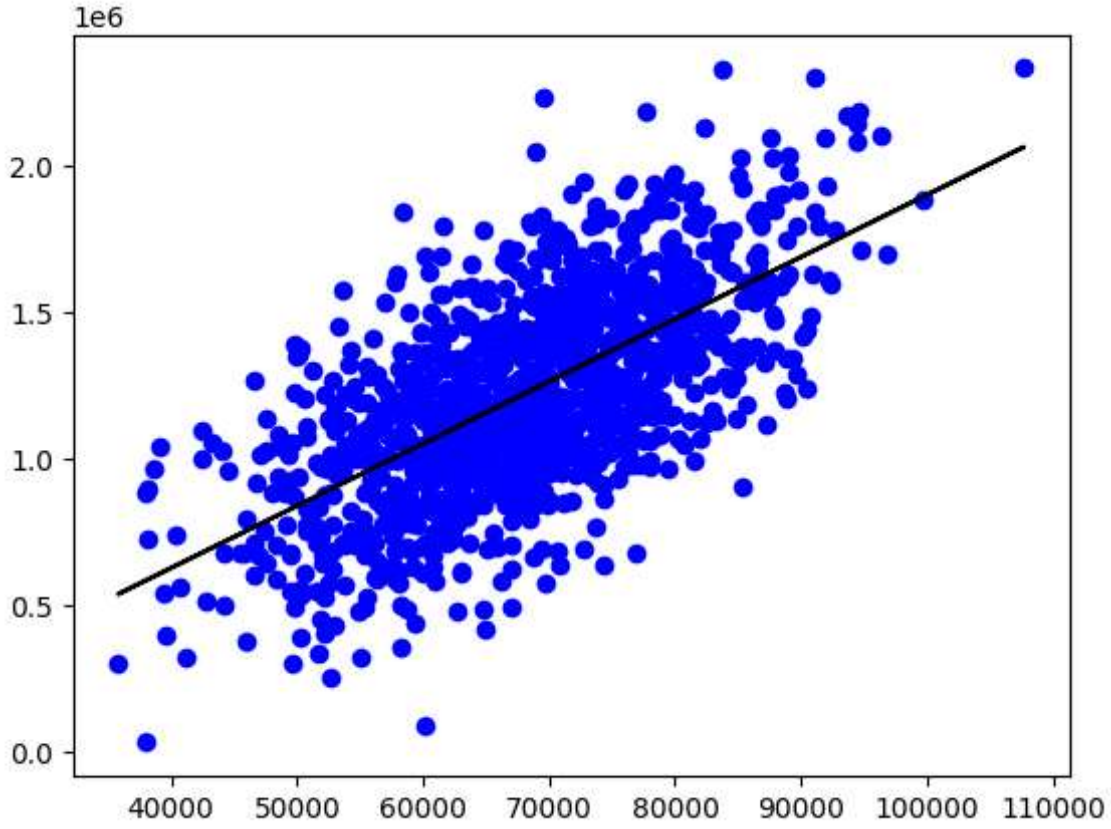
In [22]:

```
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression: ",regr.score(x_test,y_test))
```

```
Regression:  0.4158919191035203
```

In [23]:

```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

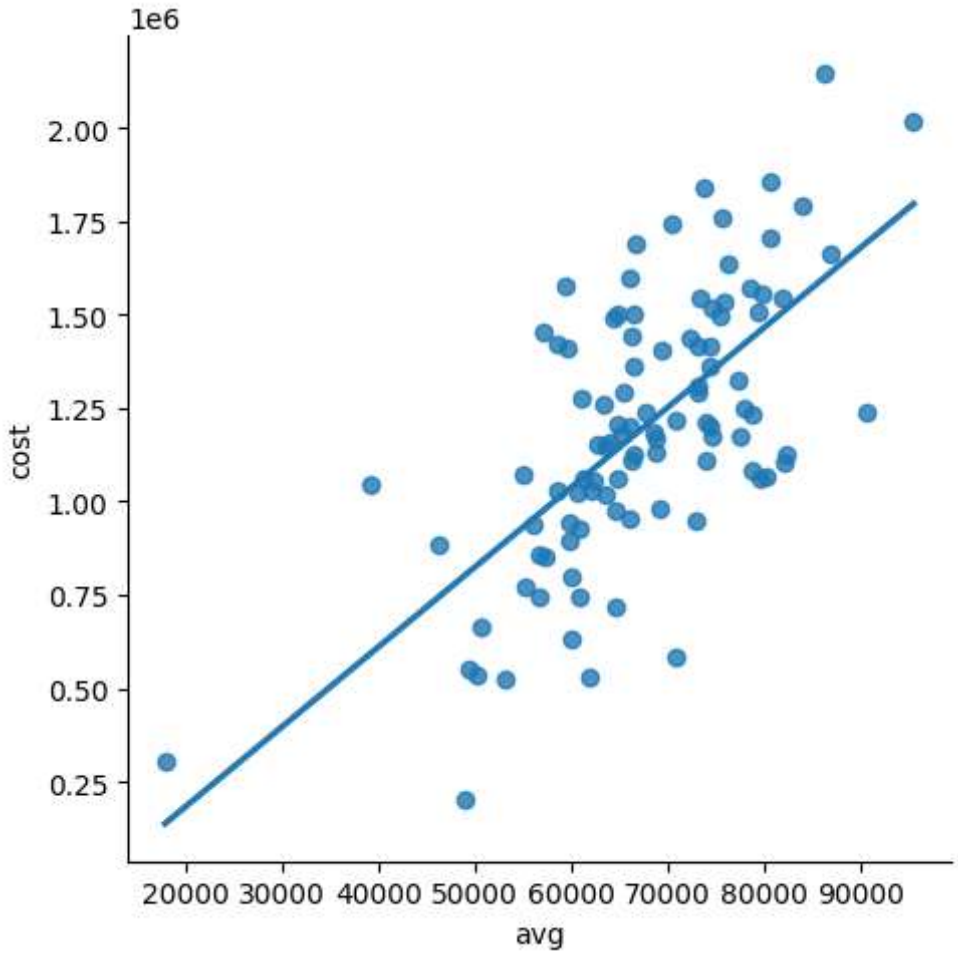


In [24]:

```
df100=df[:][:100]
sns.lmplot(x='avg',y='cost',data=df100,order=1,ci=None)
```

Out[24]:

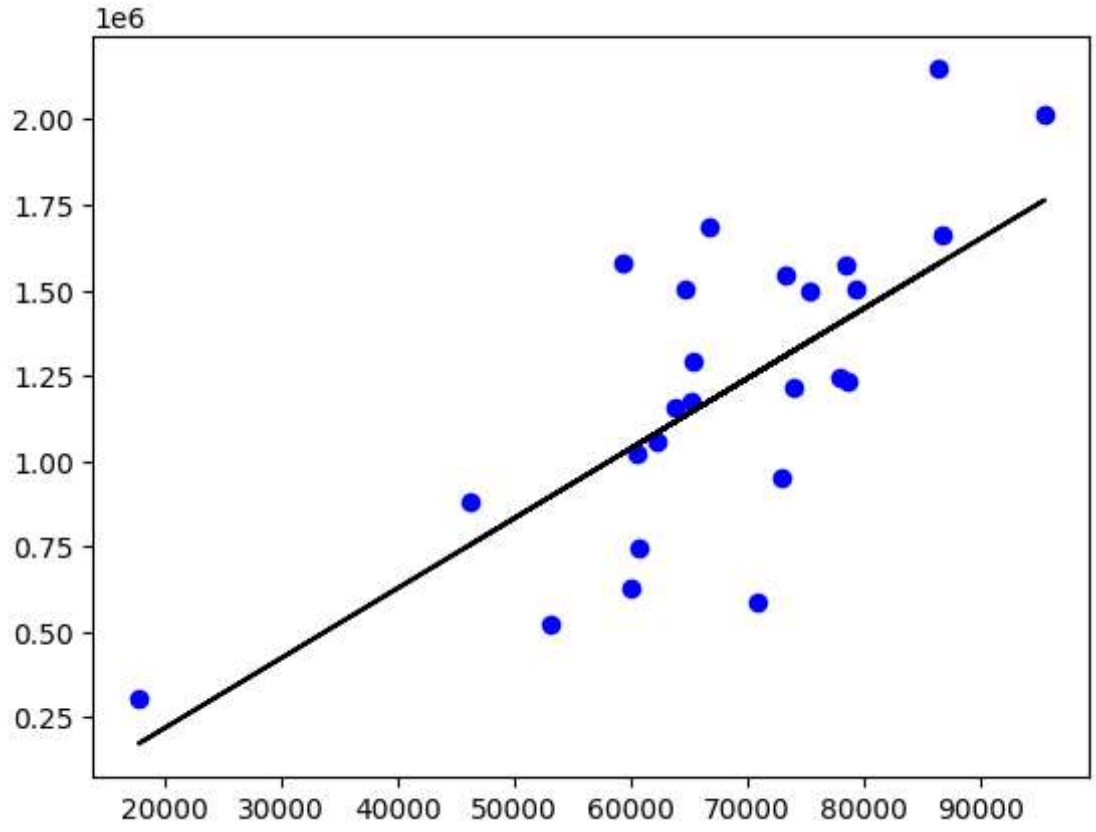
```
<seaborn.axisgrid.FacetGrid at 0x215c6156ad0>
```



In [25]:

```
df100.fillna(method='ffill',inplace=True)
x = np.array(df100['avg']).reshape(-1,1)
y = np.array(df100['cost']).reshape(-1,1)
df100.dropna(inplace=True)
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25)
regr = LinearRegression()
regr.fit(x_train,y_train)
print("regression: ",regr.score(x_test,y_test))
y_pred = regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
regression:  0.5431385856434479
```

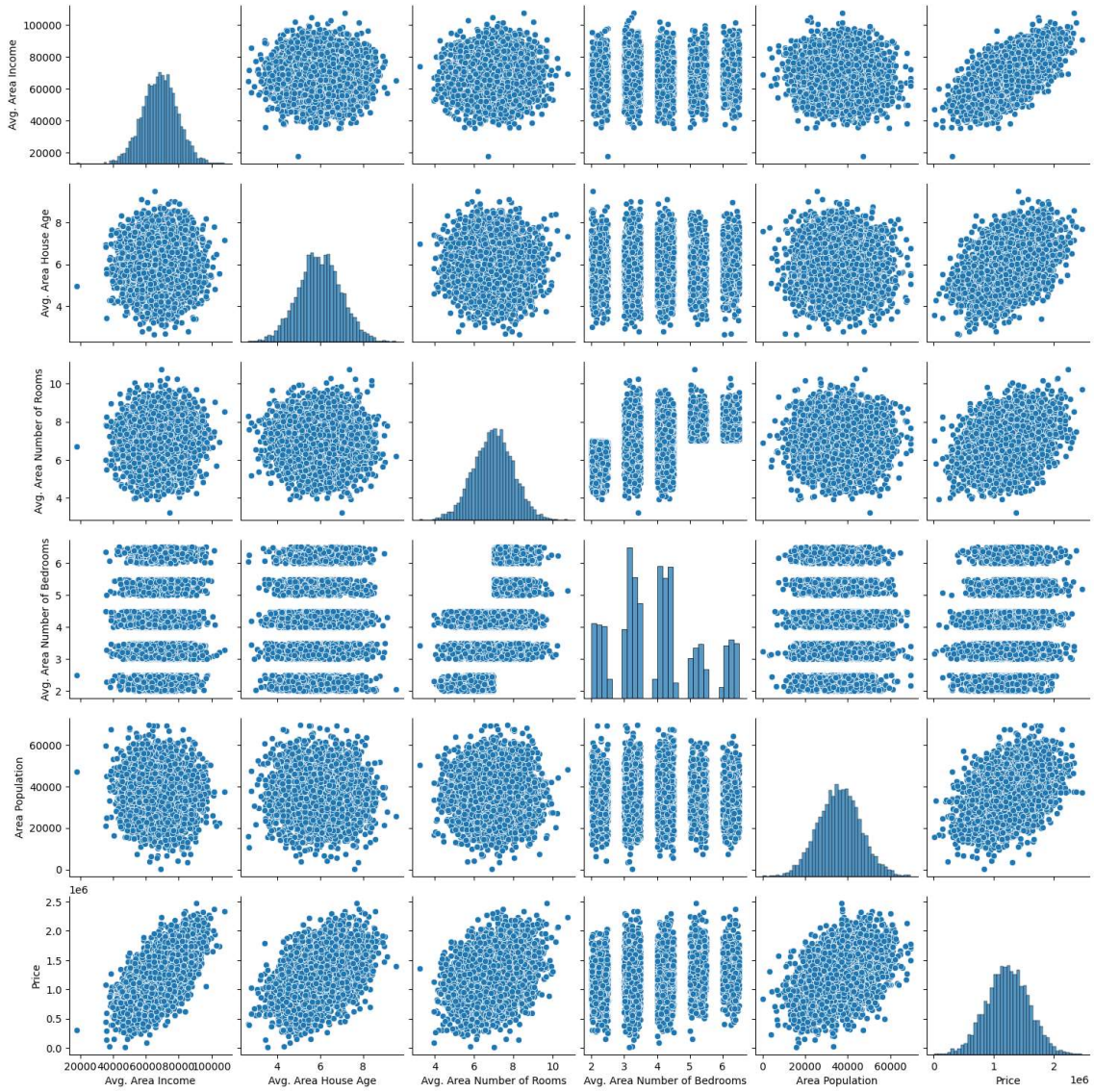


In [8]:

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

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x20ebe3f2a10>
```

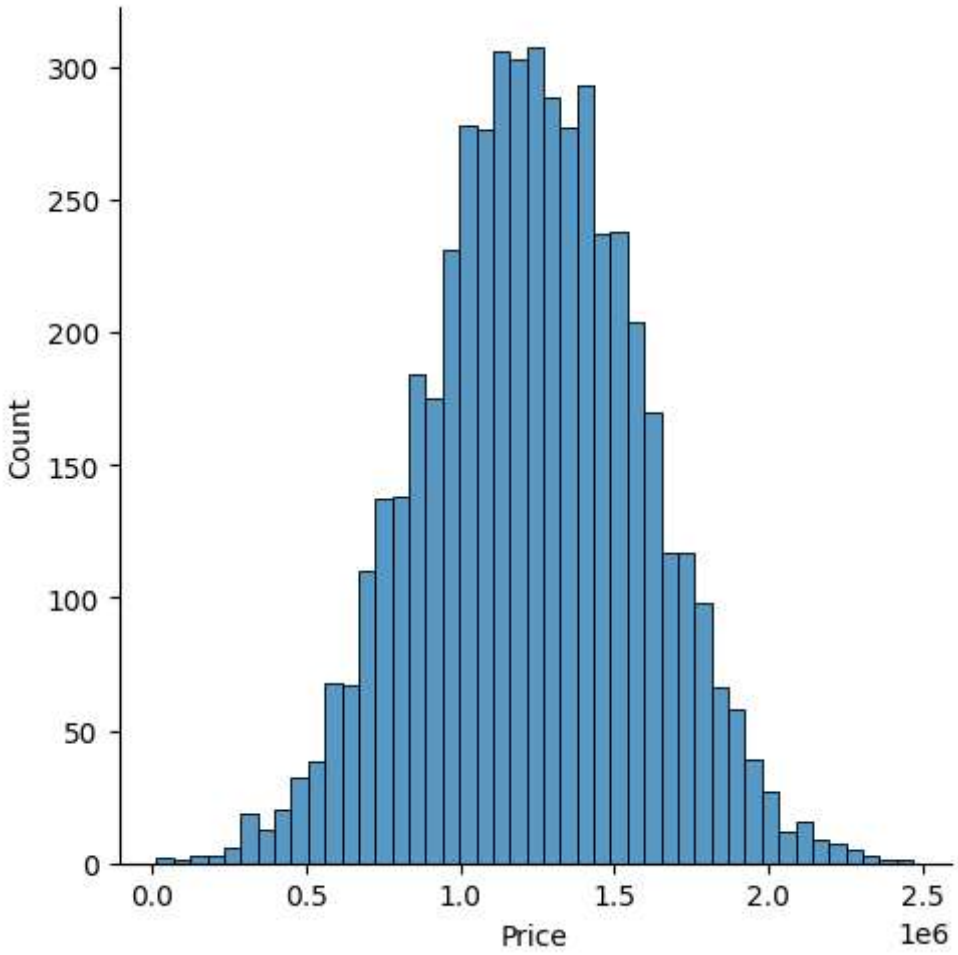


In [9]:

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

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x20ed3826f90>
```

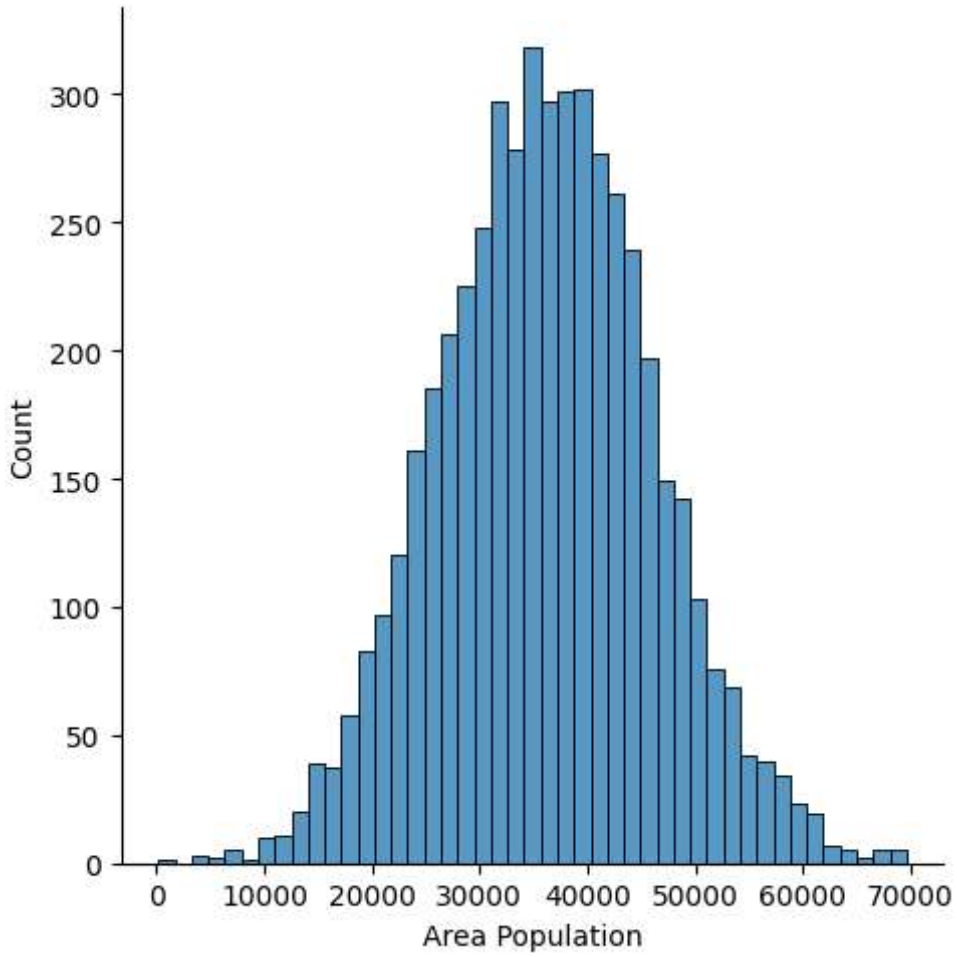


In [10]:

```
sns.displot(df['Area Population'])
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x20ed4b5d710>
```

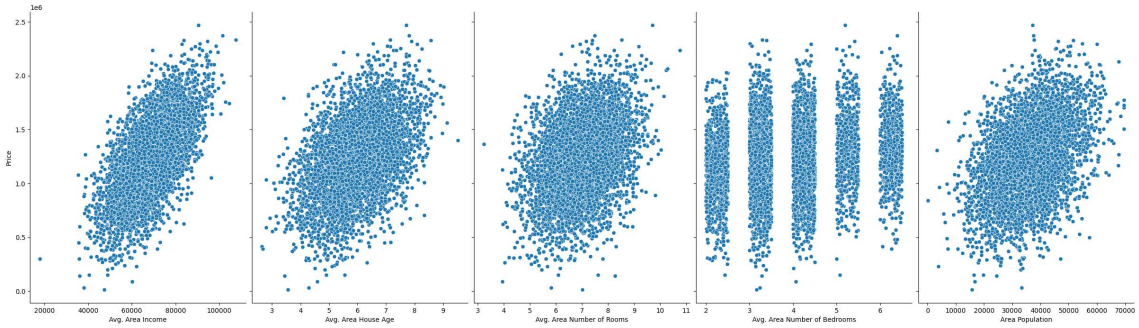


In [11]:

```
sns.pairplot(df, x_vars=['Avg. Area Income','Avg. Area House Age','Avg. Area Number of R
```

Out[11]:

```
<seaborn.axisgrid.PairGrid at 0x20ed42dba50>
```

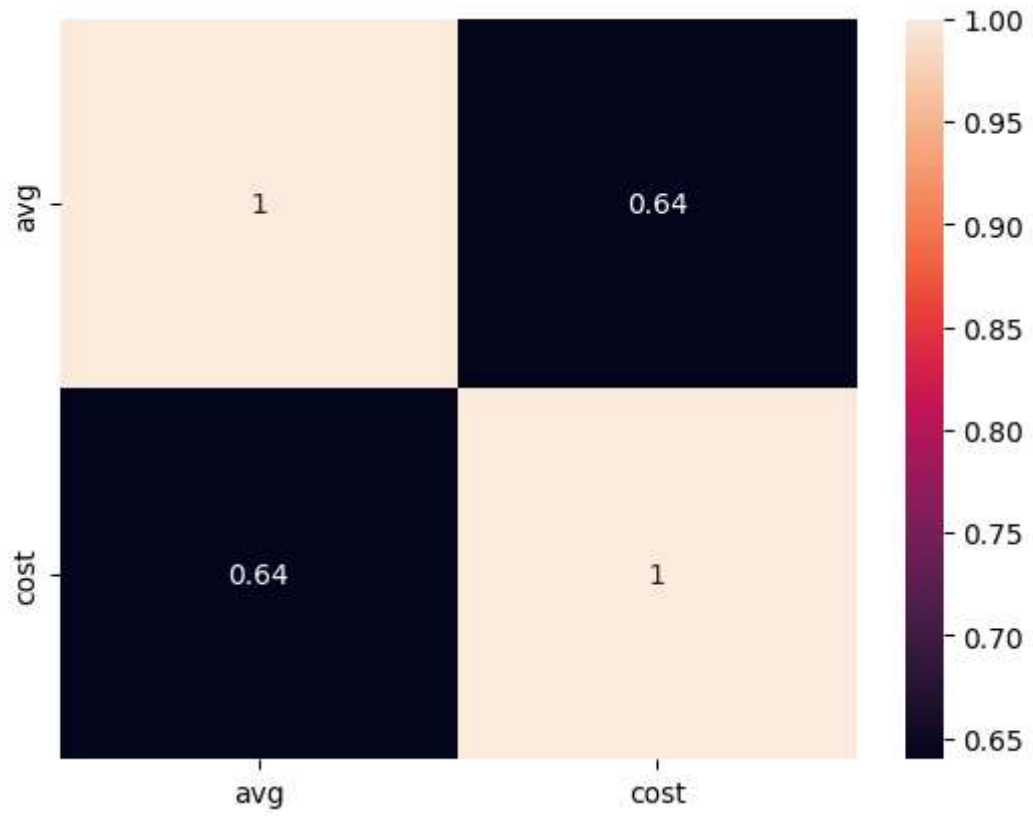


In [28]:

```
sns.heatmap(df.corr(),annot=True)
```

Out[28]:

```
<Axes: >
```



In [26]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model =LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2_Score: ",r2)
```

```
R2_Score:  0.5431385856434479
```

Conclusion:

In this dataset having minimal amount of data so LinearRegression is same for both Normal data and Minim al amount of data