In [52]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

In [2]:
```
df=pd.read_csv(r"F:\dataset\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	Addr
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry 674\nLaurabury 37
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Vi Suite 079\nL Kathleen,
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizat Stravenue\nDanielto WI 064
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPC 44
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\n AE 09
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\n AP 30153-7
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, 8489\nAPC 42991-3
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Gar Suite 076\nJoshuala VA
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPC 73
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ric Apt. 509\nEast H NV

5000 rows × 7 columns

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
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 [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.columns
```

Out[5]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

In [6]:
```
vf=df.drop(['Address'],axis=1, inplace=True)
```

In [7]:
```
df
```

Out[7]:

	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
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05
...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06

5000 rows × 6 columns

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

Out[8]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price'],
      dtype='object')
```

In [9]:
```
# Putting feature variable to X
X=df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]
```

In [10]:
```
X
```

Out[10]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
0	79545.458574	5.682861	7.009188	4.09	23086.800503
1	79248.642455	6.002900	6.730821	3.09	40173.072174
2	61287.067179	5.865890	8.512727	5.13	36882.159400
3	63345.240046	7.188236	5.586729	3.26	34310.242831
4	59982.197226	5.040555	7.839388	4.23	26354.109472
...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035
4996	78491.275435	6.999135	6.576763	4.02	25616.115489
4997	63390.686886	7.250591	4.805081	2.13	33266.145490
4998	68001.331235	5.534388	7.130144	5.44	42625.620156
4999	65510.581804	5.992305	6.792336	4.07	46501.283803

5000 rows × 5 columns

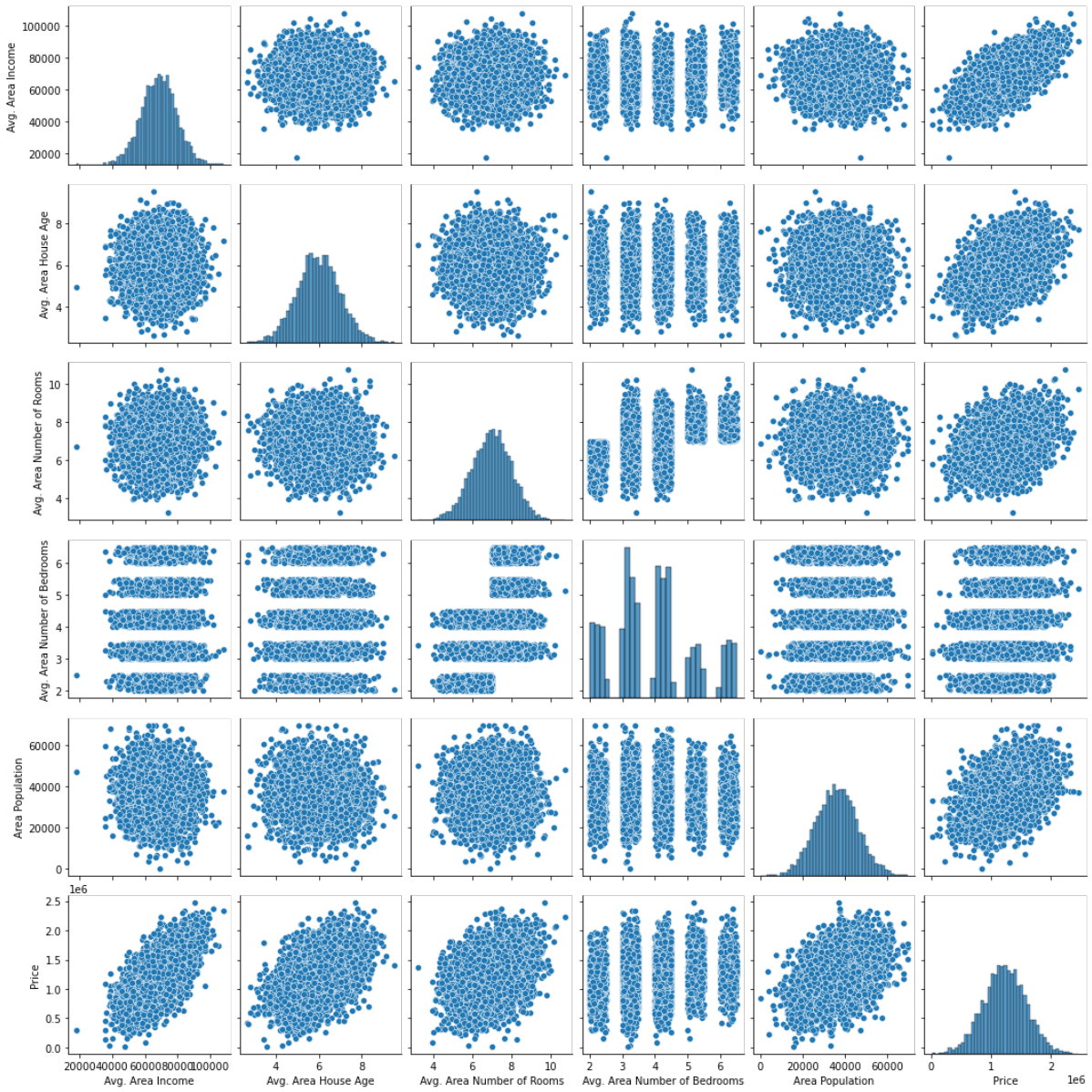
In [11]:
```
# Putting response variable to y
y=df['Price']
```

In [12]:
```
y
```

Out[12]:
```
0       1.059034e+06
1       1.505891e+06
2       1.058988e+06
3       1.260617e+06
4       6.309435e+05
            ...
4995    1.060194e+06
4996    1.482618e+06
4997    1.030730e+06
4998    1.198657e+06
4999    1.298950e+06
Name: Price, Length: 5000, dtype: float64
```

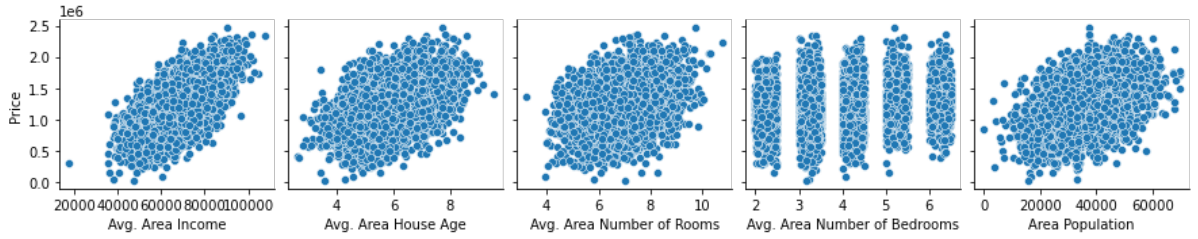
In [13]:
```
# Let's plot a pair plot of all variables in our dataframe
sns.pairplot(df)
```

Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x2422440fdc0>
```



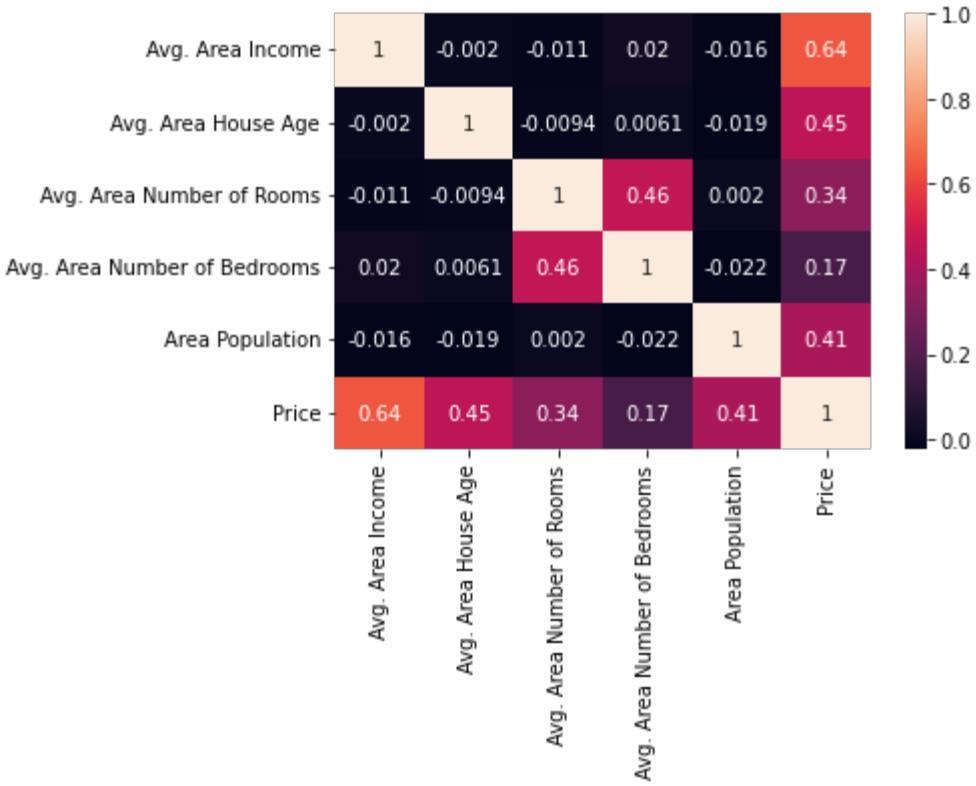
In [15]:
```
# Visualise the relationship between the features and the response using scatterplo
sns.pairplot(df,x_vars=['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Numbe
        'Avg. Area Number of Bedrooms', 'Area Population'],y_vars=['Price'])
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x2422c6b2b50>
```



In [16]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[16]:
```
<AxesSubplot:>
```



In [17]:
```
#random_state is the seed used by the random number generator, it can be any intege
from sklearn.model_selection import train_test_split
```

In [18]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,train_size=0.7 ,test_size = 0.3,
```

In [19]:
```
X_train
```

Out[19]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
2416	80238.585161	4.990994	7.017304	3.22	34271.102344
2417	60062.695634	4.169137	7.383503	3.24	45347.932064
2513	66862.876919	7.484642	6.233823	3.49	15325.648451
1698	65543.338541	3.945932	7.424297	6.38	28939.038840
3322	71328.913882	5.870775	6.011423	2.50	26738.549644
...	...	...	...	...	...
3335	86249.993070	6.155403	7.967184	4.39	43154.838627
1099	76048.372319	6.642757	7.658409	6.43	22469.522532
2514	83638.116931	7.013590	7.001637	4.29	24565.976806
3606	76637.583898	5.839368	6.620744	2.18	36236.514677
2575	65244.876417	7.729031	7.009954	6.03	39757.704309

3500 rows × 5 columns

In [20]:
```
X_test
```

Out[20]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
3566	61958.143055	6.068642	8.106395	4.14	39953.168018
4252	79055.652196	6.947016	7.509679	5.39	41724.909276
1918	64989.783962	5.162645	7.462080	6.06	34819.428301
4111	71901.333527	6.721604	6.075581	4.30	36966.474661
1471	60377.607115	3.987010	7.035224	4.06	29114.288143
...	...	...	...	...	...
570	64595.711552	5.857966	6.075798	4.02	29668.876686
4853	77053.135363	5.274005	8.940146	6.09	37397.680630
768	85673.306204	6.592507	6.368279	2.14	33110.560197
2085	53657.291997	5.992907	6.087748	2.37	38994.082726
4836	71208.269301	5.300326	6.077989	4.01	25696.361741

1500 rows × 5 columns

In [21]:
```
y_train
```

Out[21]:
```
2416    1.130844e+06
2417    7.845034e+05
2513    1.028494e+06
1698    7.498472e+05
3322    1.062105e+06
           ...
3335    1.749820e+06
1099    1.205750e+06
2514    1.569600e+06
3606    1.405060e+06
2575    1.495384e+06
Name: Price, Length: 3500, dtype: float64
```

In [22]:
```
y_test.shape
```

Out[22]:
```
(1500,)
```

In [23]:
```
y_train.values
```

Out[23]:
```
array([1130844.02936265,  784503.35791889, 1028493.60145884, ...,
       1569600.44563595, 1405059.64167422, 1495384.00366879])
```

In [24]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3500, 5)
(1500, 5)
(3500,)
(1500,)
```

In [ ]:
```
# Importing RFE and LinearRegression
```

In [25]:
```
from sklearn.linear_model import LinearRegression
```

In [26]:
```
# Representing LinearRegression as lr(Creating LinearRegression Object)
lr=LinearRegression()
```

In [27]:
```
lr.fit(X_train.values,y_train.values)
```

Out[27]:
```
LinearRegression()
```

In [28]:
```
lr.intercept_
```

Out[28]:
```
-2623728.373508366
```

In [29]:
```
# Let's see the coefficient
coeffi=pd.DataFrame(lr.coef_,X_test.columns,columns=["coefficenct"])
coeffi.round()
```

Out[29]:

	coefficenct
Avg. Area Income	22.0
Avg. Area House Age	165019.0
Avg. Area Number of Rooms	119429.0
Avg. Area Number of Bedrooms	2444.0
Area Population	15.0

In [53]:
```
# Making predictions using the model
y_pred=lr.predict(X_test)
y_pred
```

Out[53]:
```
array([1296103.71031501, 1763747.44657968, 1056152.77938735, ...,
       1579150.99568055,  846605.26162811,  907163.13233001])
```

In [31]:
```
from sklearn.metrics import mean_squared_error,r2_score
```

In [32]:
```
mse=mean_squared_error(y_test,y_pred)
r_sq=r2_score(y_test,y_pred)
```

In [33]:
```
print('mean_squared_error:',mse)
print('r2:',r_sq)
```

```
mean_squared_error: 9831074697.740602
r2: 0.9199287959786
```

In [ ]:
```
# using statsmodel
```

In [34]:
```
import statsmodels.api as sm
```

In [35]:
```
#Unlike SKLearn, statsmodels don't automatically fit a constant,
#so you need to use the method sm.add_constant(X) in order to add a constant.
X_train_sm=sm.add_constant(X_train)
lm=sm.OLS(y_train,X_train_sm).fit()
```

In [36]:
```
lm.params
```

Out[36]:
```
const                          -2.623728e+06
Avg. Area Income                2.151420e+01
Avg. Area House Age             1.650187e+05
Avg. Area Number of Rooms       1.194286e+05
Avg. Area Number of Bedrooms    2.443933e+03
Area Population                 1.516993e+01
dtype: float64
```

In [37]:
```
print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.917
Model:                            OLS   Adj. R-squared:                  0.917
Method:                 Least Squares   F-statistic:                     7739.
Date:                Tue, 02 Jan 2024   Prob (F-statistic):               0.00
Time:                        18:46:07   Log-Likelihood:                -45329.
No. Observations:                3500   AIC:                         9.067e+04
Df Residuals:                    3494   BIC:                         9.071e+04
Df Model:                           5
Covariance Type:            nonrobust
================================================================================
                                 coef    std err          t      P>|t|      [0.02
--------------------------------------------------------------------------------
const                      -2.624e+06   2.06e+04   -127.298      0.000   -2.66e+0
Avg. Area Income              21.5142      0.161    133.438      0.000      21.19
Avg. Area House Age          1.65e+05   1731.158     95.323      0.000    1.62e+0
Avg. Area Number of Rooms    1.194e+05   1928.552     61.927      0.000    1.16e+0
Avg. Area Number of Bedrooms 2443.9333   1585.938      1.541      0.123    -665.52
Area Population               15.1699      0.173     87.777      0.000      14.83
==============================================================================
Omnibus:                        3.606   Durbin-Watson:                   2.046
Prob(Omnibus):                  0.165   Jarque-Bera (JB):                3.281
Skew:                           0.011   Prob(JB):                        0.194
Kurtosis:                       2.852   Cond. No.                     9.37e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly sp
[2] The condition number is large, 9.37e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [38]:
```
X_train.drop(["Avg. Area Number of Bedrooms"],axis=1,inplace=True)
```

In [39]:
```
X_test.drop(["Avg. Area Number of Bedrooms"],axis=1,inplace=True)
```

In [40]:
```
X_train
```

Out[40]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Area Population
2416	80238.585161	4.990994	7.017304	34271.102344
2417	60062.695634	4.169137	7.383503	45347.932064
2513	66862.876919	7.484642	6.233823	15325.648451
1698	65543.338541	3.945932	7.424297	28939.038840
3322	71328.913882	5.870775	6.011423	26738.549644
...	...	...	...	...
3335	86249.993070	6.155403	7.967184	43154.838627
1099	76048.372319	6.642757	7.658409	22469.522532
2514	83638.116931	7.013590	7.001637	24565.976806
3606	76637.583898	5.839368	6.620744	36236.514677
2575	65244.876417	7.729031	7.009954	39757.704309

3500 rows × 4 columns

In [41]:
```
lr.fit(X_train,y_train)
```

Out[41]:
```
LinearRegression()
```

In [42]:
```
lr.intercept_
```

Out[42]:
```
-2623603.8683603564
```

In [43]:
```
# Making predictions using the model
y_pred=lr.predict(X_test)
y_pred
```

Out[43]:
```
array([1296103.71031501, 1763747.44657968, 1056152.77938735, ...,
       1579150.99568055,  846605.26162811,  907163.13233001])
```

In [44]:
```
mse=mean_squared_error(y_test,y_pred)
mse
```

Out[44]:
```
9823431323.317856
```

In [45]:
```
r=r2_score(y_pred,y_test)
r
```

Out[45]:
```
0.9109451655588731
```

In [49]:
```
# Actual and Predicted
c = [i for i in range(1,1501,1)] # generating index
fig = plt.figure(figsize=(12,8))
plt.plot(c,y_test, color="blue", linewidth=2.5, linestyle="-") #Plotting Actual
plt.plot(c,y_pred, color="red",  linewidth=2.5, linestyle="-") #Plotting predicted
fig.suptitle('Actual and Predicted', fontsize=15)              # Plot heading
plt.xlabel('Index', fontsize=18)                               # X-label
plt.ylabel('Housing Price', fontsize=16)                       # Y-label
```

Out[49]:
```
Text(0, 0.5, 'Housing Price')
```