In [42]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as seabornInstance
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
%matplotlib inline
```

In [43]:
```
#read file
df=pd.read_csv("USA_Housing.csv")
df.head()
```

Out[43]:

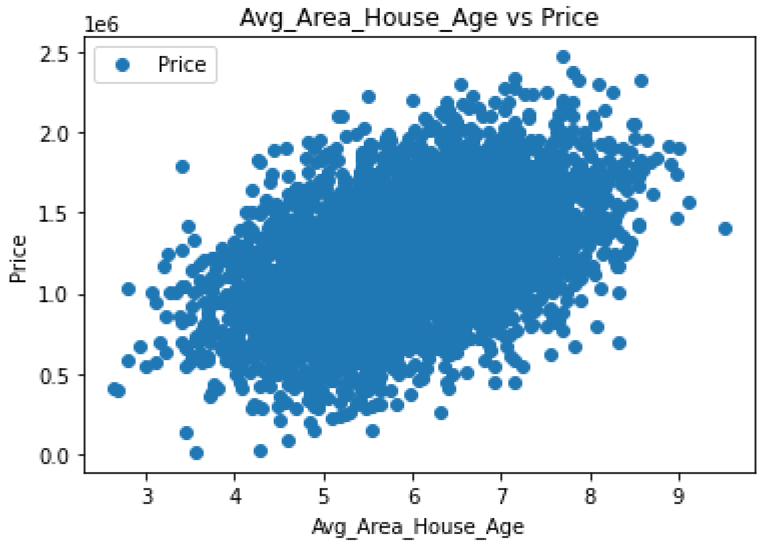
	Avg_Area_Income	Avg_Area_House_Age	Avg_Area_Number_of_Rooms	Avg_Area_Number_of_Bec
0	79545.45857	5.682861	7.009188	
1	79248.64245	6.002900	6.730821	
2	61287.06718	5.865890	8.512727	
3	63345.24005	7.188236	5.586729	
4	59982.19723	5.040555	7.839388	

In [44]:
```
#To see the statistical details of the dataset
df.describe()
```

Out[44]:

	Avg_Area_Income	Avg_Area_House_Age	Avg_Area_Number_of_Rooms	Avg_Area_Number_of
count	5000.000000	5000.000000	5000.000000	5
mean	68583.108984	5.977222	6.987792	
std	10657.991214	0.991456	1.005833	
min	17796.631190	2.644304	3.236194	
25%	61480.562390	5.322283	6.299250	
50%	68804.286405	5.970429	7.002902	
75%	75783.338665	6.650808	7.665871	
max	107701.748400	9.519088	10.759588	

In [45]:
```
#plot our data points on a 2-D graph to eyeball our dataset and see if we can mar
df.plot(x='Avg_Area_House_Age', y='Price', style='o')
plt.title('Avg_Area_House_Age vs Price')
plt.xlabel('Avg_Area_House_Age')
plt.ylabel('Price')
plt.show()
```



In [46]:
```
# the data into “attributes” and “labels”.
#Attributes are the independent variables while labels are dependent variables wh
X = df['Avg_Area_House_Age'].values.reshape(-1,1)
y = df['Price'].values.reshape(-1,1)
```

In [47]:
```
#we split 20% of the data to the training set while 80% of the data to test set u
#The test_size variable is where we actually specify the proportion of the test s
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8, random_s
```

In [48]:
```
#to train our algorithm. For that, we need to import LinearRegression class
regressor = LinearRegression()
regressor.fit(X_train, y_train) #training the algorithm
```

Out[48]:
```
LinearRegression()
```

In [49]:
```
#To retrieve the intercept:
print(regressor.intercept_)
#For retrieving the slope:
print(regressor.coef_)
```

```
[309418.81589309]
[[155235.07217097]]
```

In [50]:
```
#we will use our test data and see how accurately our algorithm predicts
y_pred = regressor.predict(X_test)
```

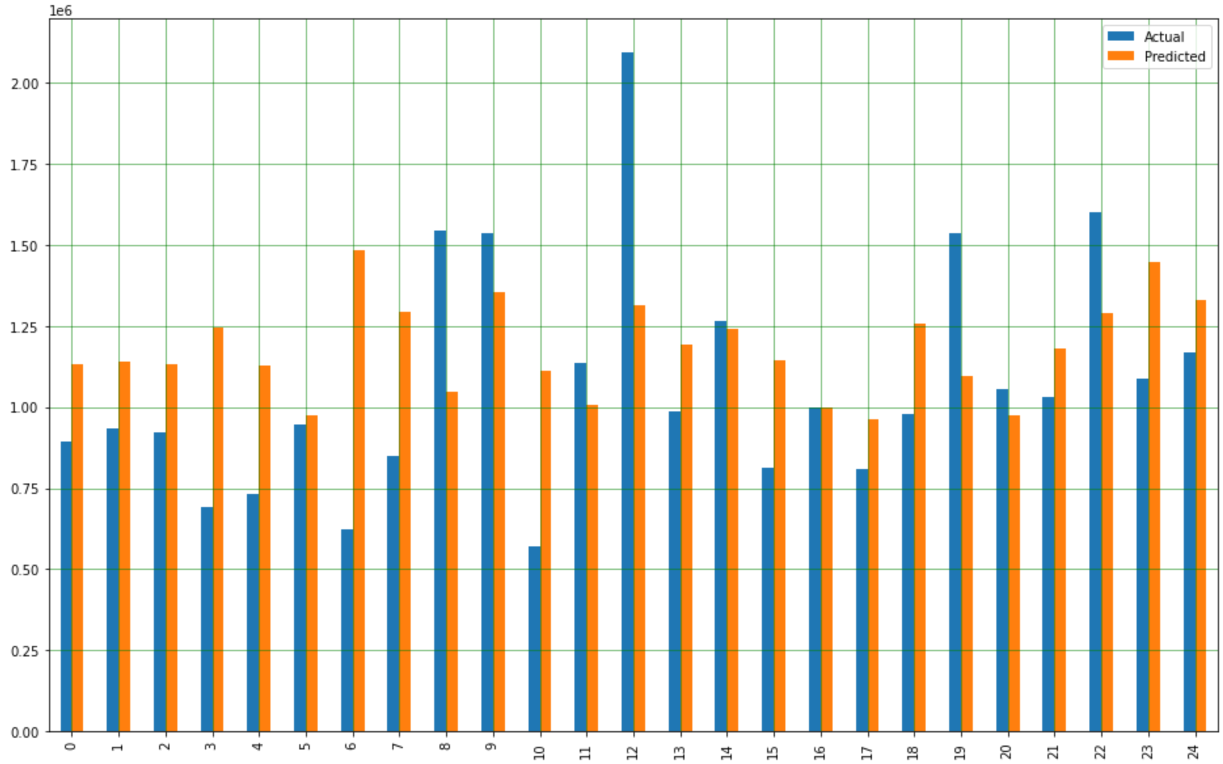
In [51]:
```
#compare the actual output values for X_test with the predicted values
df = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
df
```

Out[51]:

	Actual	Predicted
0	8.942511e+05	1.132117e+06
1	9.329794e+05	1.139098e+06
2	9.207479e+05	1.132215e+06
3	6.918549e+05	1.245093e+06
4	7.327332e+05	1.128760e+06
...	...	...
3995	9.792828e+05	1.227820e+06
3996	1.182888e+06	1.023557e+06
3997	1.423025e+06	1.277973e+06
3998	9.907257e+05	1.036107e+06
3999	1.086829e+06	1.167474e+06

4000 rows × 2 columns

In [52]:
```
#visualize comparison result as a bar graph
df1 = df.head(25)
df1.plot(kind='bar',figsize=(16,10))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```



In [ ]:

In [ ]: