In [16]:
```
!pip install scikit-learn
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scikitlearn as sklearn
```

```
Requirement already satisfied: scikit-learn in ./anaconda3/lib/python3.10/si
te-packages (1.2.1)
Requirement already satisfied: numpy>=1.17.3 in ./anaconda3/lib/python3.10/s
ite-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: joblib>=1.1.1 in ./anaconda3/lib/python3.10/s
ite-packages (from scikit-learn) (1.1.1)
Requirement already satisfied: scipy>=1.3.2 in ./anaconda3/lib/python3.10/si
te-packages (from scikit-learn) (1.10.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in ./anaconda3/lib/pytho
n3.10/site-packages (from scikit-learn) (2.2.0)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[16], line 5
      3 import pandas as pd
      4 import matplotlib.pyplot as plt
----> 5 import scikitlearn as sklearn

ModuleNotFoundError: No module named 'scikitlearn'
```

In [2]:
```
df=pd.read_csv("/Users/myyntiimac/Desktop/House_data.csv")
df.head()
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	flo
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	

5 rows × 21 columns

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

Out[3]:
```
id               False
date             False
price            False
bedrooms         False
bathrooms        False
sqft_living      False
sqft_lot         False
floors           False
waterfront       False
view             False
condition        False
grade            False
sqft_above       False
sqft_basement    False
yr_built         False
yr_renovated     False
zipcode          False
lat              False
long             False
sqft_living15    False
sqft_lot15       False
dtype: bool
```

We see in this data set , no null value and all values is numerical , so we can go for model build, As we build Single linear regressein model , so we need two variable s where one is dependent anouther is independent By understanding the data we decide that price is dependable variable and and bedrooms take as independent variable

In [19]:
```
# now define the variable
bedroom=df["bedrooms"]
price=df["price"]
```

In [20]:
```
# now convert this individual attribute to numpy array
#By converting the data into NumPy arrays, we ensure compatibility with thes
#reshape ,reshape the 'bedroom' array to have id array and  a single feature
x=np.array(bedroom).reshape(-1, 1)
x
```

Out[20]:
```
array([[3],
       [3],
       [2],
       ...,
       [2],
       [3],
       [2]])
```

In [21]:
```
y=np.array(price)
y
```

Out[21]:
```
array([221900., 538000., 180000., ..., 402101., 400000., 325000.])
```

In [22]:
```
#Now we  are going to split the data for training and test
# Fo this we need to import train_test_split() from scikitlearn.modelselecti
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x,y,test_size=0.25, random_s
```

In [23]:
```
xtrain
```

Out[23]:
```
array([[1],
       [3],
       [3],
       ...,
       [3],
       [4],
       [4]])
```

In [24]:
```
xtest
```

Out[24]:
```
array([[2],
       [4],
       [2],
       ...,
       [4],
       [3],
       [3]])
```

In [25]:
```
ytrain
```

Out[25]:
```
array([420850., 335000., 587100., ..., 431000., 411000., 699900.])
```

In [26]:
```
 ytest
```

Out[26]:
```
array([ 297000., 1580000.,  562100., ...,  774950.,  372500.,  599995.])
```

In [27]:
```
#Now we import the LR algo from sklearn.linear_model and define it for train
from sklearn.linear_model import LinearRegression
LR = LinearRegression()
LR.fit(xtrain, ytrain)
```

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

In [29]:
```
# now we build the model , and test the model with test data and predict the
y_predict=LR.predict(xtest)
y_predict
```

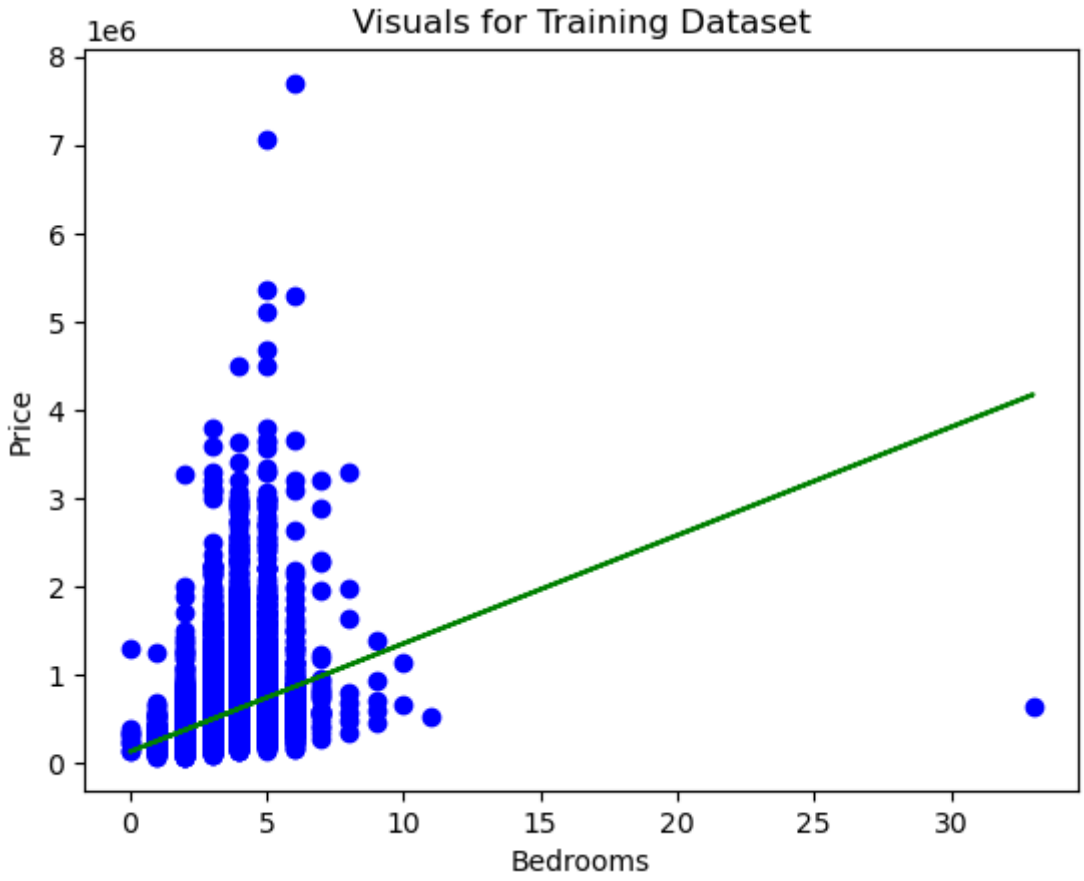
Out[29]:
```
array([374077.11696055, 619174.14320538, 374077.11696055, ...,
       619174.14320538, 496625.63008297, 496625.63008297])
```

now we are going to test our model accuracy for this we need to test , before that we can see our model by visualization

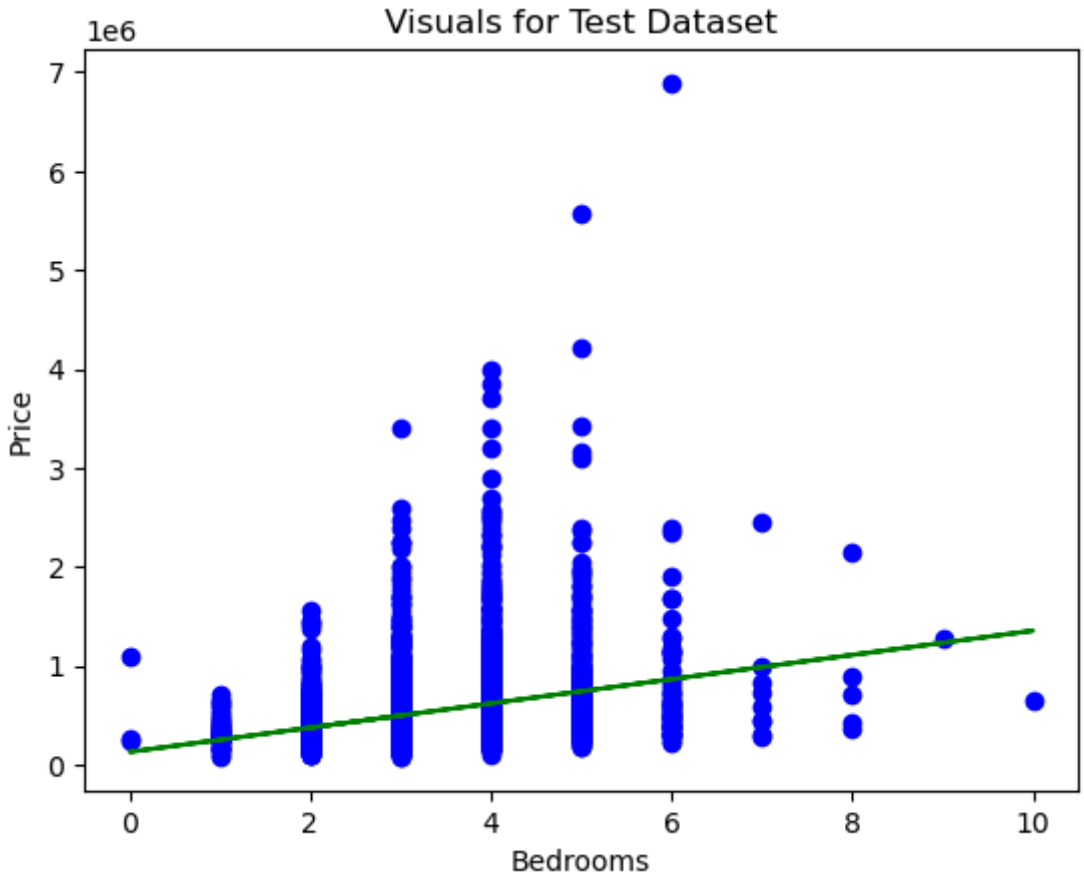
create a scatter plot with the training data points as red dots

and plot the predicted values from the regression model as a blue line.

In [30]:
```
plt.scatter(xtrain, ytrain, color= 'blue')
plt.plot(xtrain, LR.predict(xtrain), color = 'green')
plt.title ("Visuals for Training Dataset")
plt.xlabel("Bedrooms")
plt.ylabel("Price")
plt.show()
```



In [31]:
```
#similarly we can see the scatterplot and regression plot by using test data
#and predicted reggression line by our model with x test data
#25% of data
plt.scatter(xtest, ytest, color= 'blue')
plt.plot(xtest, LR.predict(xtest), color = 'green')
plt.title ("Visuals for Test Dataset")
plt.xlabel("Bedrooms")
plt.ylabel("Price")
plt.show()
```



In [33]:
```
#now we find the coeffecient of the build model and intercept or constant mo
LR.coef_
```

Out[33]:
```
array([122548.51312242])
```

In [34]:
```
LR.intercept_
```

Out[34]:
```
128980.09071571735
```

In [35]:
```
#Test with unknown data with model coefficient and intercept with 10 bedroom
Y_unknown=(128980.09071571735+122548.51312242*10)

Y_unknown
```

Out[35]:
```
1354465.2219399174
```

In [36]:
```
#now we are going to calculate bias and variance to check the model overfitt
#Bias check with train data
Bias=LR.score(xtrain, ytrain)
Bias
```

Out[36]:
```
0.09895262216814216
```

In [37]:
```
# Varince test score calculate by test data
varience=LR.score(xtest, ytest)
varience
```

Out[37]:
```
0.08286189035217228
```

Insight:with a low bias and low variance, it can be inferred that the model has a good balance between capturing the underlying patterns and generalizing to new data. This

suggests that the model has a good accuracy and is likely to perform well on unseen data.