In [61]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import statistics
import datetime
from statistics import mode,mean
os.chdir("C:/Data Science/Real Estate")

data = pd.read_csv('RealEstateAU_1000_Samples_upd.csv')
data.info()
font1 = {'family':'serif','color':'blue','size':20}
sns.set_style('whitegrid')
plt.figure(figsize=(14,7))
plt.title('Price by all Property Type',fontdict = font1)
sns.barplot(x= data['property_type'] ,y= data['price'])


data1 = pd.read_csv('RealEstateAU_1000_Samples_upd.csv')
data = data1[data1['property_type']=='Apartment']
data.info()
font1 = {'family':'serif','color':'blue','size':20}
sns.set_style('whitegrid')
plt.figure(figsize=(14,7))
plt.title('Price by City Property Type Apartment',fontdict = font1)
plt.xlabel("City")
plt.ylabel("Price")
sns.barplot(x= data['city'] ,y=data['index'])

data = data1[data1['property_type']=='House']
data.info()
font1 = {'family':'serif','color':'blue','size':20}
sns.set_style('whitegrid')
plt.figure(figsize=(14,7))
plt.title('Price by City and Property Type House',fontdict = font1)
plt.xlabel("City")
plt.ylabel("Price")
sns.barplot(x= data['city'] ,y=data['price'])



data = data1[data1['property_type']=='Residential Land']
data.info()
font1 = {'family':'serif','color':'blue','size':20}
sns.set_style('whitegrid')
plt.figure(figsize=(14,7))
plt.title('Price by City and Property Type Residential Land',fontdict = font1)
plt.xlabel("City")
plt.ylabel("Price")
sns.barplot(x= data['city'] ,y=data['price'])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 467 entries, 0 to 466
Data columns (total 28 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
```

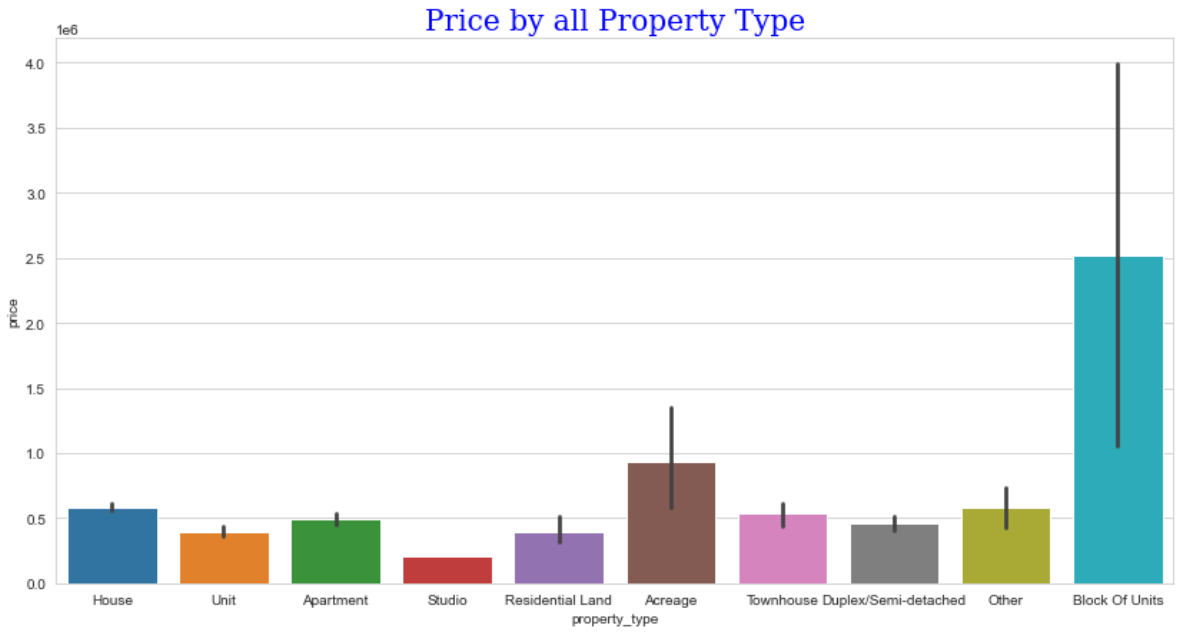
```
 0   index            467 non-null    int64
 1   TID              467 non-null    int64
 2   breadcrumb       467 non-null    object
 3   category_name    467 non-null    object
 4   property_type    467 non-null    object
 5   building_size    135 non-null    object
 6   land_size        241 non-null    object
 7   preferred_size   279 non-null    object
 8   open_date        100 non-null    object
 9   listing_agency   467 non-null    object
 10  price            467 non-null    int64
 11  location_number  467 non-null    int64
 12  location_type    467 non-null    object
 13  location_name    467 non-null    object
 14  address          460 non-null    object
 15  address_1        460 non-null    object
 16  city             467 non-null    object
 17  state            467 non-null    object
 18  zip_code         467 non-null    int64
 19  phone            467 non-null    object
 20  latitude         0 non-null      float64
 21  longitude        0 non-null      float64
 22  product_depth    467 non-null    object
 23  bedroom_count    442 non-null    float64
 24  bathroom_count   442 non-null    float64
 25  parking_count    442 non-null    float64
 26  RunDate          467 non-null    object
 27  Count            467 non-null    float64
dtypes: float64(6), int64(5), object(17)
memory usage: 102.3+ KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 112 entries, 4 to 457
Data columns (total 28 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   index            112 non-null    int64
 1   TID              112 non-null    int64
 2   breadcrumb       112 non-null    object
 3   category_name    112 non-null    object
 4   property_type    112 non-null    object
 5   building_size    32 non-null     object
 6   land_size        23 non-null     object
 7   preferred_size   32 non-null     object
 8   open_date        14 non-null     object
 9   listing_agency   112 non-null    object
 10  price            112 non-null    int64
 11  location_number  112 non-null    int64
 12  location_type    112 non-null    object
 13  location_name    112 non-null    object
 14  address          112 non-null    object
 15  address_1        112 non-null    object
 16  city             112 non-null    object
 17  state            112 non-null    object
 18  zip_code         112 non-null    int64
 19  phone            112 non-null    object
 20  latitude         0 non-null      float64
 21  longitude        0 non-null      float64
 22  product_depth    112 non-null    object
 23  bedroom_count    112 non-null    float64
 24  bathroom_count   112 non-null    float64
 25  parking_count    112 non-null    float64
```

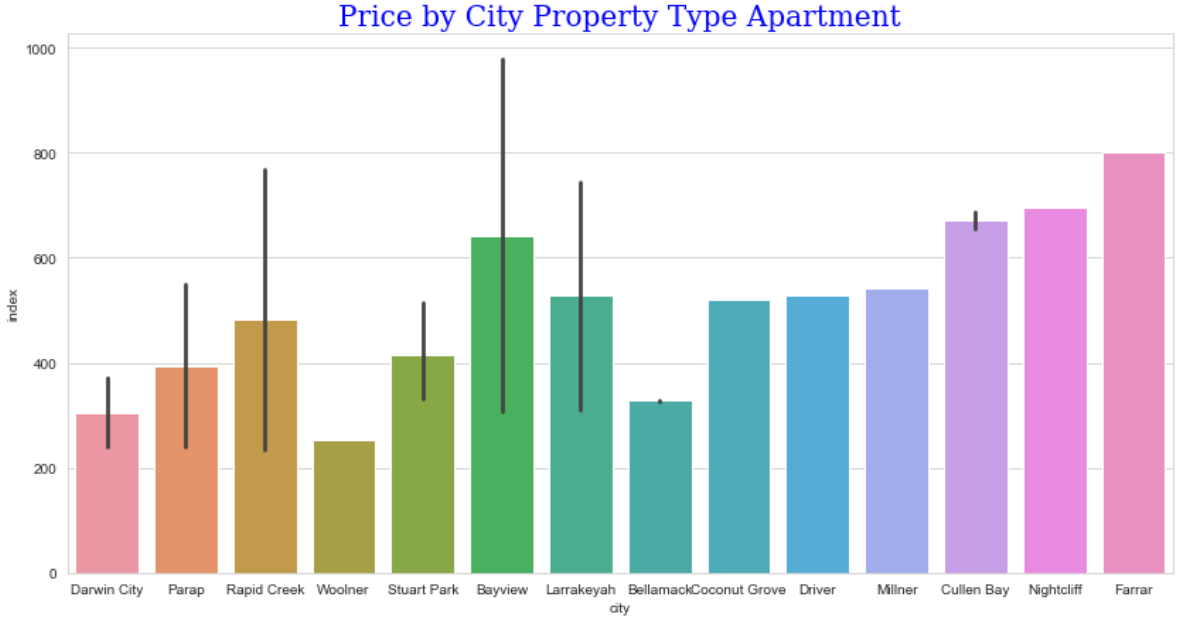
```
 26  RunDate          112 non-null    object
 27  Count            112 non-null    float64
dtypes: float64(6), int64(5), object(17)
memory usage: 25.4+ KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 186 entries, 0 to 465
Data columns (total 28 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   index            186 non-null    int64
 1   TID              186 non-null    int64
 2   breadcrumb       186 non-null    object
 3   category_name    186 non-null    object
 4   property_type    186 non-null    object
 5   building_size    66 non-null     object
 6   land_size        137 non-null    object
 7   preferred_size   141 non-null    object
 8   open_date        50 non-null     object
 9   listing_agency   186 non-null    object
 10  price            186 non-null    int64
 11  location_number  186 non-null    int64
 12  location_type    186 non-null    object
 13  location_name    186 non-null    object
 14  address          183 non-null    object
 15  address_1        183 non-null    object
 16  city             186 non-null    object
 17  state            186 non-null    object
 18  zip_code         186 non-null    int64
 19  phone            186 non-null    object
 20  latitude         0 non-null      float64
 21  longitude        0 non-null      float64
 22  product_depth    186 non-null    object
 23  bedroom_count    186 non-null    float64
 24  bathroom_count   186 non-null    float64
 25  parking_count    186 non-null    float64
 26  RunDate          186 non-null    object
 27  Count            186 non-null    float64
dtypes: float64(6), int64(5), object(17)
memory usage: 42.1+ KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25 entries, 95 to 397
Data columns (total 28 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   index            25 non-null     int64
 1   TID              25 non-null     int64
 2   breadcrumb       25 non-null     object
 3   category_name    25 non-null     object
 4   property_type    25 non-null     object
 5   building_size    0 non-null      object
 6   land_size        22 non-null     object
 7   preferred_size   22 non-null     object
 8   open_date        2 non-null      object
 9   listing_agency   25 non-null     object
 10  price            25 non-null     int64
 11  location_number  25 non-null     int64
 12  location_type    25 non-null     object
 13  location_name    25 non-null     object
 14  address          23 non-null     object
 15  address_1        23 non-null     object
 16  city             25 non-null     object
```

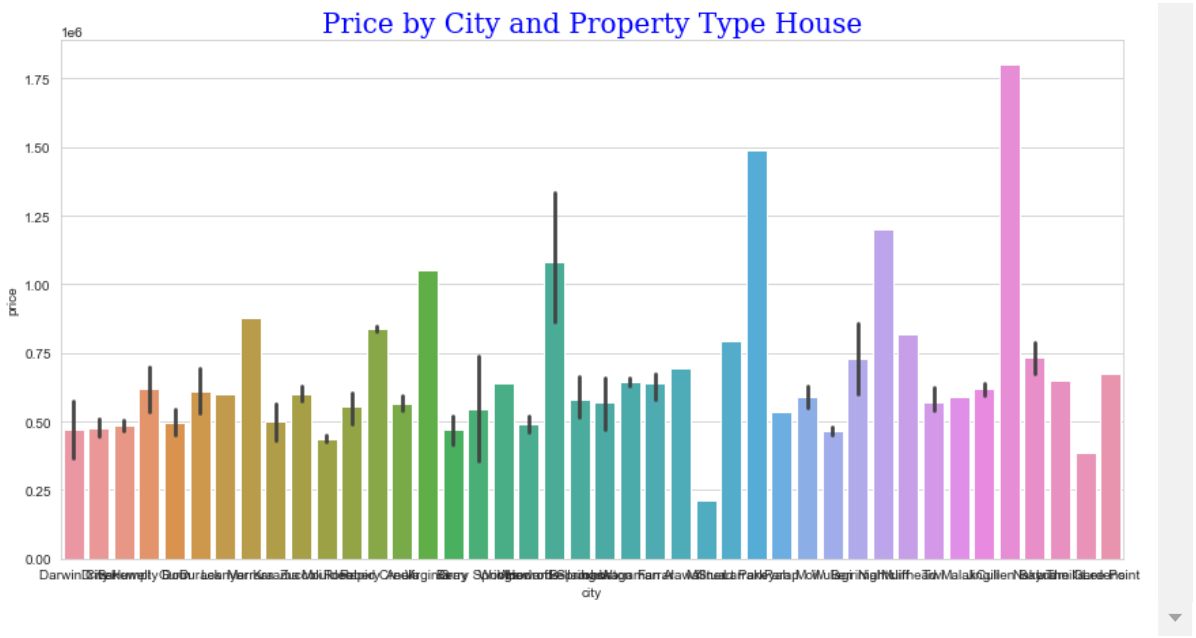
```
 17  state             25 non-null     object
 18  zip_code          25 non-null     int64
 19  phone             25 non-null     object
 20  latitude          0 non-null      float64
 21  longitude         0 non-null      float64
 22  product_depth     25 non-null     object
 23  bedroom_count     0 non-null      float64
 24  bathroom_count    0 non-null      float64
 25  parking_count     0 non-null      float64
 26  RunDate           25 non-null     object
 27  Count             25 non-null     float64
dtypes: float64(6), int64(5), object(17)
memory usage: 5.7+ KB
```

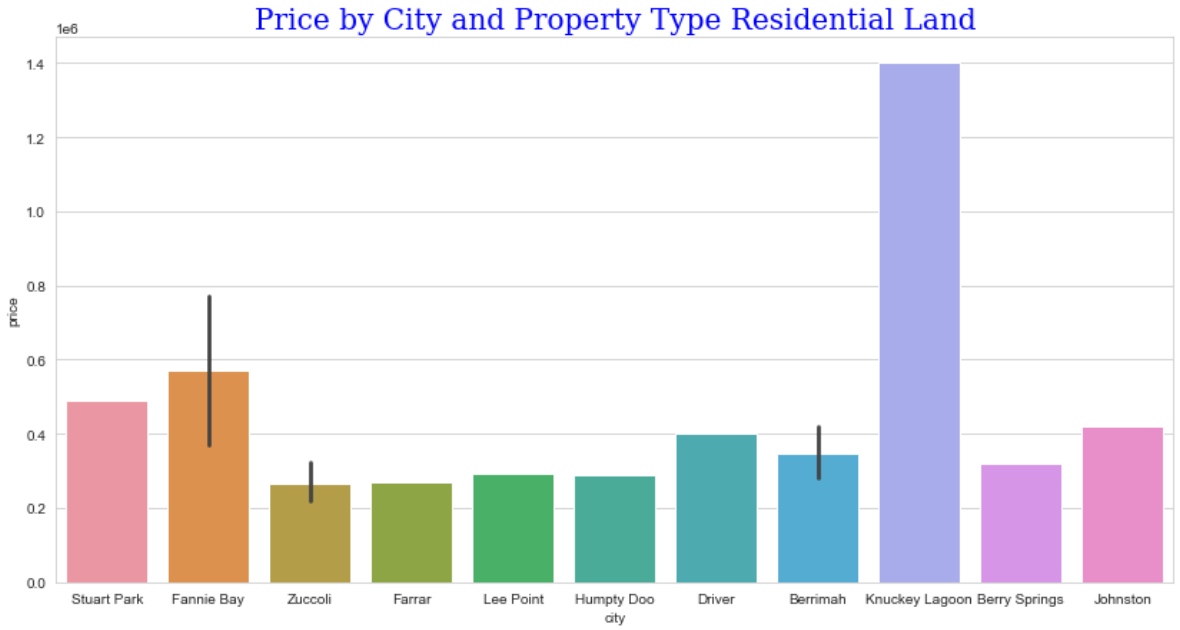
Out[61]:

```
<AxesSubplot:title={'center':'Price by City and Property Type Residential La
nd'}, xlabel='city', ylabel='price'>
```






Price by City and Property Type House
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
price
city


Price by City and Property Type Residential Land
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
price
Stuart Park
Fannie Bay
Zuccoli
Farrar
Lee Point
Humpty Doo
Driver
Berrimah
Knuckey Lagoon
Berry Springs
Johnston
city

In [ ]:

In [ ]:

In [ ]:

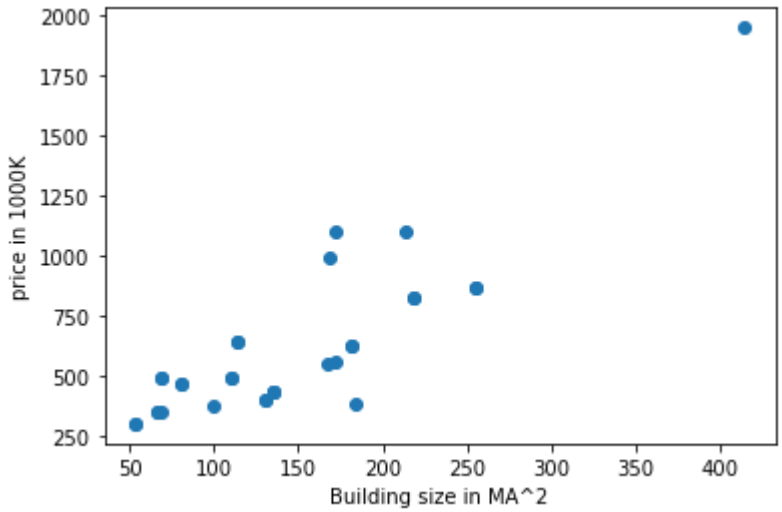
In [20]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
from statistics import mode,mean
os.chdir("C:/Data Science")
data = pd.read_csv('RealEstateAU_1000_Samples.csv')
data.head()
data.columns
data.info()
data = data[data['property_type']=='Apartment']
data = data[["building_size","price"]]

print(data.head())
plt.scatter(data["building_size"],data["price"]/1000)
plt.xlabel("Building size in MA^2")
plt.ylabel("price in 1000K")
plt.show()
num = int(len(data)*0.8)
#training
train = data[:num]
#testing
test = data[num:]
print("Data:",len(data))
print("Train:",len(train))
print("Test:",len(test))

#Main function to find the coefficients of line:
def simple_linear_regression(input_feature, output):
 Xi = input_feature
 Yi = output

#Total number of data points:
 n = len(Xi)
#X bar:
 Xi_mean = Xi.mean( )

#Y bar:
 Yi_mean = Yi.mean ( )
#Sum of X:
 S_Xi = (Xi).sum( )
#Sum of Y:

 S_Yi = (Yi).sum( )
#Sum of (X*Y) multiplied by n:

 S_XiYi = ((Xi*Yi).sum( ) )*n
#Sum of X*Sum of Y:

 S_Xi_S_Yi = S_Xi*S_Yi
#Sum of (X*X) multiplied by n:
 S_XiXi = ((Xi*Xi).sum( ) )*n

#Square of sum of X:
 S_Xi_Square = S_Xi*S_Xi
#Slope:

 slope = (S_XiYi- S_Xi_S_Yi) / (S_XiXi-S_Xi_Square)
#Intercept:
```

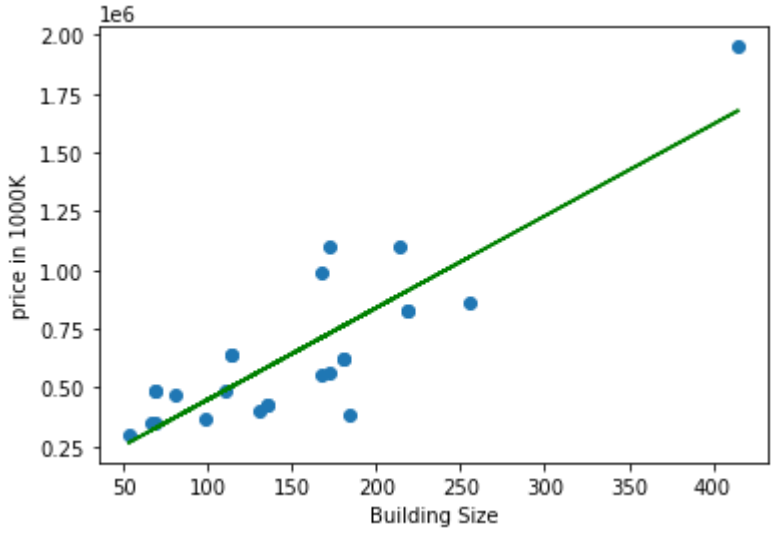
```
 intercept = Yi_mean - slope * Xi_mean
 return slope, intercept

#Training the model with train data:
#Finding the coefficients of best fit line:
actual_slope, actual_intercept = simple_linear_regression(train["building_size"], train["pr

print ("Slope: " , actual_slope)
print ("Intercept: " , actual_intercept)

#Plot the regression Line with training data:
plt . scatter(train [ "building_size"], train["price"])
plt. plot(train["building_size"], actual_slope*train ["building_size"]+actual_intercept, co
plt.xlabel("Building Size")
plt . ylabel("price in 1000K")
plt. show( )

#Define the prediction function:
def get_regression_prediction(input_features, slope, intercept) :
 predicted_value = actual_slope*input_features + actual_intercept
 return predicted_value

#Predicting values based on prediction function:
my_buiding_size = 12

estimated_price = get_regression_prediction(my_buiding_size, actual_slope, actual_intercept
print ("Estimated Price: ", estimated_price)

#Predicting values for the whole dataset:
y_pred = get_regression_prediction (data["building_size" ], actual_slope, actual_intercept)
print (y_pred)

#Create a dataframe for Actual and Predicted values:
A_P_data = pd . DataFrame ({"Actual" : data["price"] , "Predicted" :y_pred})
print (A_P_data.head())
#Plot the bar graph for actual and predicted values:
A_P_data.plot(kind='bar',figsize=(12, 6))
#A_P_data.head(10).plot(kind='bar',figsize=(12, 6))
plt.title('Actual Vs Predicted  for property type Apartment with Linaer Regression')
plt.show( )


#Error calculation using Residual Sum of Squares:
def residual_sum_of_squares (input_feature, output, slope, intercept):
    prediction = slope*input_feature + intercept
    residual = (output - prediction)
    RSS = (residual*residual).sum()
    return (RSS )
print("RSs: ",residual_sum_of_squares(test["building_size"],test["price"],actual_slope,actu
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 135 entries, 0 to 134
Data columns (total 28 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   index            135 non-null    int64
 1   TID              135 non-null    int64
 2   breadcrumb       135 non-null    object
 3   category_name    135 non-null    object
 4   property_type    135 non-null    object
```

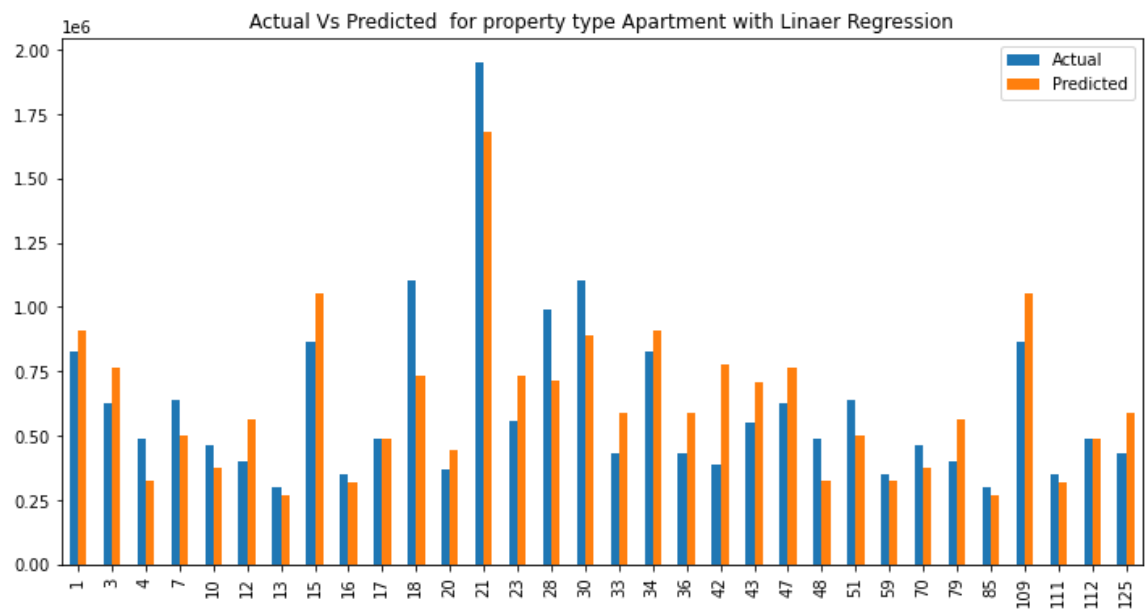
```
 5   building_size    135 non-null    float64
 6   land_size        75 non-null     object
 7   preferred_size   134 non-null    object
 8   open_date        26 non-null     object
 9   listing_agency   135 non-null    object
 10  price            135 non-null    int64
 11  location_number  135 non-null    int64
 12  location_type    135 non-null    object
 13  location_name    135 non-null    object
 14  address          134 non-null    object
 15  address_1        134 non-null    object
 16  city             135 non-null    object
 17  state            135 non-null    object
 18  zip_code         135 non-null    int64
 19  phone            135 non-null    object
 20  latitude         0 non-null      float64
 21  longitude        0 non-null      float64
 22  product_depth    135 non-null    object
 23  bedroom_count    135 non-null    int64
 24  bathroom_count   135 non-null    int64
 25  parking_count    135 non-null    int64
 26  RunDate          135 non-null    object
 27  Count            135 non-null    int64
dtypes: float64(3), int64(9), object(16)
memory usage: 29.7+ KB
    building_size   price
1           218.0  825000
3           181.0  625000
4            69.0  490000
7           114.0  640000
10           81.0  465000
```




```
Data: 32
Train: 25
Test: 7
Slope:  3904.106811957714
Intercept:  57974.174290533294
```




```
Estimated Price:  104823.45603402586
1      9.090695e+05
3      7.646175e+05
4      3.273575e+05
7      5.030424e+05
10     3.742068e+05
12     5.655081e+05
13     2.648918e+05
15     1.053521e+06
16     3.156452e+05
17     4.874259e+05
18     7.297148e+05
20     4.444807e+05
21     1.678179e+06
23     7.294805e+05
28     7.138641e+05
30     8.922037e+05
33     5.850286e+05
34     9.090695e+05
36     5.850286e+05
42     7.763298e+05
43     7.099600e+05
47     7.646175e+05
48     3.273575e+05
51     5.030424e+05
59     3.273575e+05
70     3.742068e+05
79     5.655081e+05
85     2.648918e+05
109    1.053521e+06
111    3.156452e+05
112    4.874259e+05
125    5.850286e+05
Name: building_size, dtype: float64
    Actual      Predicted
1   825000  909069.459297
3   625000  764617.507255
4   490000  327357.544316
7   640000  503042.350854
10  465000  374206.826059
```



RSs: 97892765298.35864

In [ ]:

In [ ]: