Basic Exploratory Data Analysis on Lagos House Prices

- The dataset contains 5336 records of referral properties arround 7 districts in Lagos. The Dataset was sourced from a Nigeria real estate company website

In [1]:
```
# import the neccessary libraries
import numpy as np
import pandas as pd

# import visuals Python Libraries
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
#Now we want to load the dataset which means to 'load and read dataset'
# We will title it as follows

lagos_houses = pd.read_csv(r'C:\Users\hp\Desktop\Data Analysis Training\Python\DataSet\lagos_house_prices_raw.csv') #th
```

In [3]:
```
lagos_houses = pd.read_csv(r'C:\Users\hp\Desktop\Data Analysis Training\Python\DataSet\lagos_house_prices_raw.csv')
lagos_houses
```

Out[3]:

	location	bed	bath	toilet	price	Property_Type	Parking_Space	Security	Electricity	Furnished	Security_Doors	CCTV	Pool	Gyn
0	yaba	1	1	2	700000.0	Mini flat	0	0	0	0	0	0	0	0
1	yaba	1	1	2	700000.0	Mini flat	0	0	0	0	0	0	0	0
2	yaba	1	1	2	650000.0	Mini flat	0	0	0	0	0	0	0	0
3	yaba	1	1	1	450000.0	Mini flat	0	0	0	0	0	0	0	0
4	yaba	3	3	4	800000.0	Detached duplex	0	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5331	ajah	1	1	2	600000.0	Mini flat	1	0	0	0	0	0	0	0
5332	ajah	2	2	2	700000.0	Mini flat	1	0	0	0	0	0	0	0
5333	ajah	4	4	5	1700000.0	Semi detached duplex	1	0	0	0	0	0	0	0
5334	ajah	1	1	2	500000.0	Mini flat	0	0	0	0	0	0	0	0
5335	ajah	4	4	5	1800000.0	Semi detached duplex	1	1	0	1	0	0	0	0

5336 rows × 15 columns

In [4]:
```
# to check the first 5 rows
lagos_houses.head()
```

Out[4]:

	location	bed	bath	toilet	price	Property_Type	Parking_Space	Security	Electricity	Furnished	Security_Doors	CCTV	Pool	Gym	BQ
0	yaba	1	1	2	700000.0	Mini flat	0	0	0	0	0	0	0	0	
1	yaba	1	1	2	700000.0	Mini flat	0	0	0	0	0	0	0	0	
2	yaba	1	1	2	650000.0	Mini flat	0	0	0	0	0	0	0	0	
3	yaba	1	1	1	450000.0	Mini flat	0	0	0	0	0	0	0	0	
4	yaba	3	3	4	800000.0	Detached duplex	0	1	0	0	0	0	0	0	

In [5]:
```
# to check the last 5 rows
lagos_houses.tail()
```

Out[5]:

	location	bed	bath	toilet	price	Property_Type	Parking_Space	Security	Electricity	Furnished	Security_Doors	CCTV	Pool	Gyn
5331	ajah	1	1	2	600000.0	Mini flat	1	0	0	0	0	0	0	
5332	ajah	2	2	2	700000.0	Mini flat	1	0	0	0	0	0	0	
5333	ajah	4	4	5	1700000.0	Semi detached duplex	1	0	0	0	0	0	0	
5334	ajah	1	1	2	500000.0	Mini flat	0	0	0	0	0	0	0	
5335	ajah	4	4	5	1800000.0	Semi detached duplex	1	1	0	1	0	0	0	

Here at this juncture what we are doing is Data inspection and Manipulation

In [7]:
```
# to check the shape
lagos_houses.shape
```

Out[7]:
```
(5336, 15)
```

In [8]:
```
# to check the columns
lagos_houses.columns
```

Out[8]:
```
Index(['location', 'bed', 'bath', 'toilet', 'price', 'Property_Type',
       'Parking_Space', 'Security', 'Electricity', 'Furnished',
       'Security_Doors', 'CCTV', 'Pool', 'Gym', 'BQ'],
      dtype='object')
```

In [9]:
```
# to check the datatypes
lagos_houses.dtypes
```

Out[9]:
```
location          object
bed                int64
bath               int64
toilet             int64
price            float64
Property_Type     object
Parking_Space      int64
Security           int64
Electricity        int64
Furnished          int64
Security_Doors     int64
CCTV               int64
Pool               int64
Gym                int64
BQ                 int64
dtype: object
```

In [10]:
```
# to check the info of the dataset
lagos_houses.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5336 entries, 0 to 5335
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   location        5336 non-null   object 
 1   bed             5336 non-null   int64  
 2   bath            5336 non-null   int64  
 3   toilet          5336 non-null   int64  
 4   price           5336 non-null   float64
 5   Property_Type   5336 non-null   object 
 6   Parking_Space   5336 non-null   int64  
 7   Security        5336 non-null   int64  
 8   Electricity     5336 non-null   int64  
 9   Furnished       5336 non-null   int64  
 10  Security_Doors  5336 non-null   int64  
 11  CCTV            5336 non-null   int64  
 12  Pool            5336 non-null   int64  
 13  Gym             5336 non-null   int64  
 14  BQ              5336 non-null   int64  
dtypes: float64(1), int64(12), object(2)
memory usage: 625.4+ KB
```

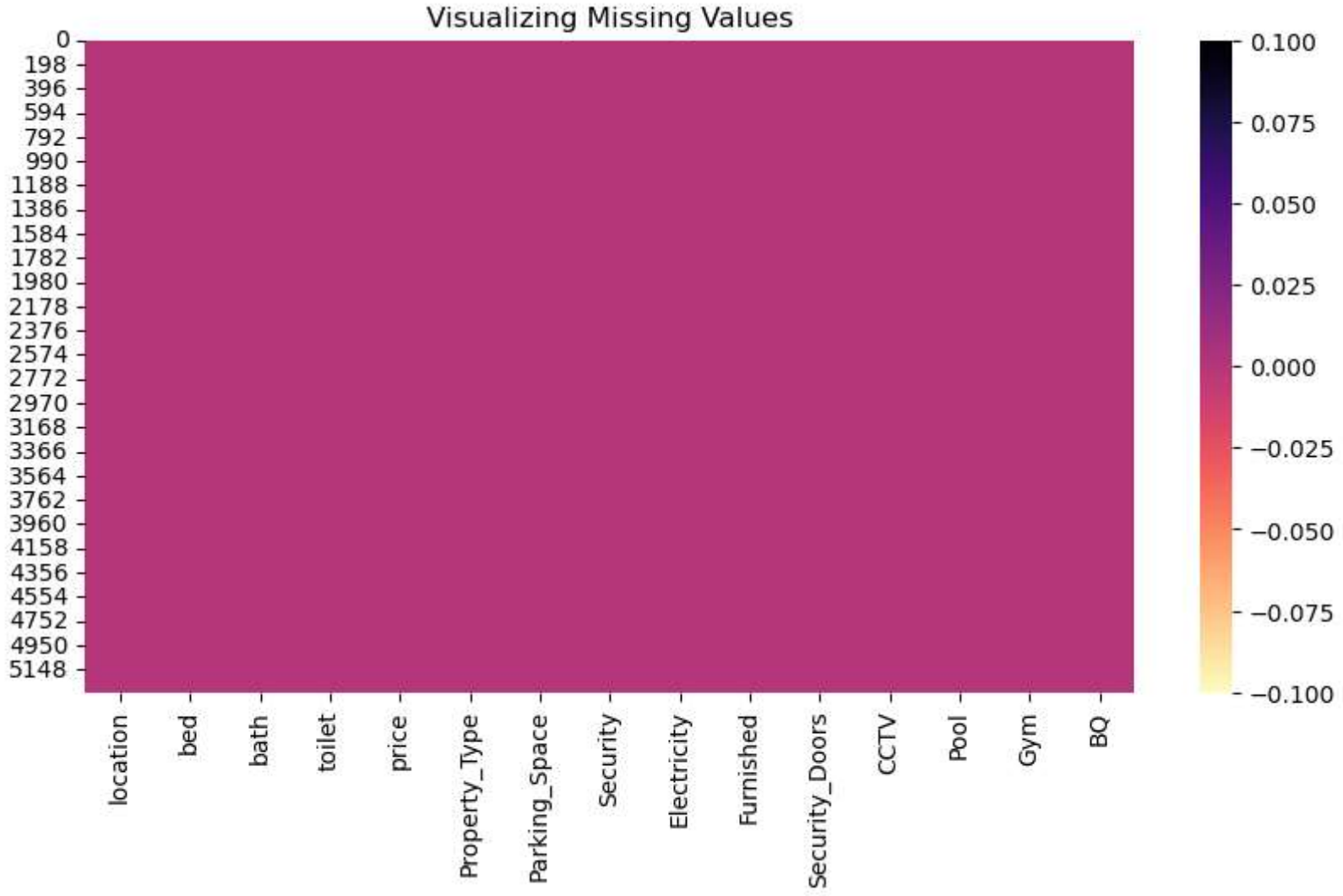
In [11]:
```
# Checking missing values or null values
lagos_houses.isna().sum()
```

Out[11]:
```
location          0
bed               0
bath              0
toilet            0
price             0
Property_Type     0
Parking_Space     0
Security          0
Electricity       0
Furnished         0
Security_Doors    0
CCTV              0
Pool              0
Gym               0
BQ                0
dtype: int64
```

In [12]:
```
# Note that there are no missing values in this data set but we can also visualize if there are or there are no missing
# visualize missing values
```

```
plt.figure(figsize = (10, 5))
plt.title('Visualizing Missing Values')
sns.heatmap(lagos_houses.isnull(), cbar = True, cmap = 'magma_r')
```

Out[12]: <AxesSubplot:title={'center':'Visualizing Missing Values'}>



In [13]:
```
# Statistical descriptive analysis of the numerical data

lagos_houses.describe().astype('int')
```

Out[13]:

	bed	bath	toilet	price	Parking_Space	Security	Electricity	Furnished	Security_Doors	CCTV	Pool	Gym	BQ
count	5336	5336	5336	5336	5336	5336	5336	5336	5336	5336	5336	5336	5336
mean	1	1	1	645566	0	0	0	0	0	0	0	0	0
std	0	0	0	469305	0	0	0	0	0	0	0	0	0
min	1	1	1	150	0	0	0	0	0	0	0	0	0
25%	1	1	1	350000	0	0	0	0	0	0	0	0	0
50%	1	1	1	500000	0	0	0	0	0	0	0	0	0
75%	1	2	2	800000	0	0	0	0	0	0	0	0	0
max	5	5	5	2450000	1	1	1	1	1	1	1	1	1

Exploratory Data Analysis: Relationship, Insight and Visualization

- Univariate Analysis
- Bivariate Analysis
- Multivariate Analysis

Univariate Analysis

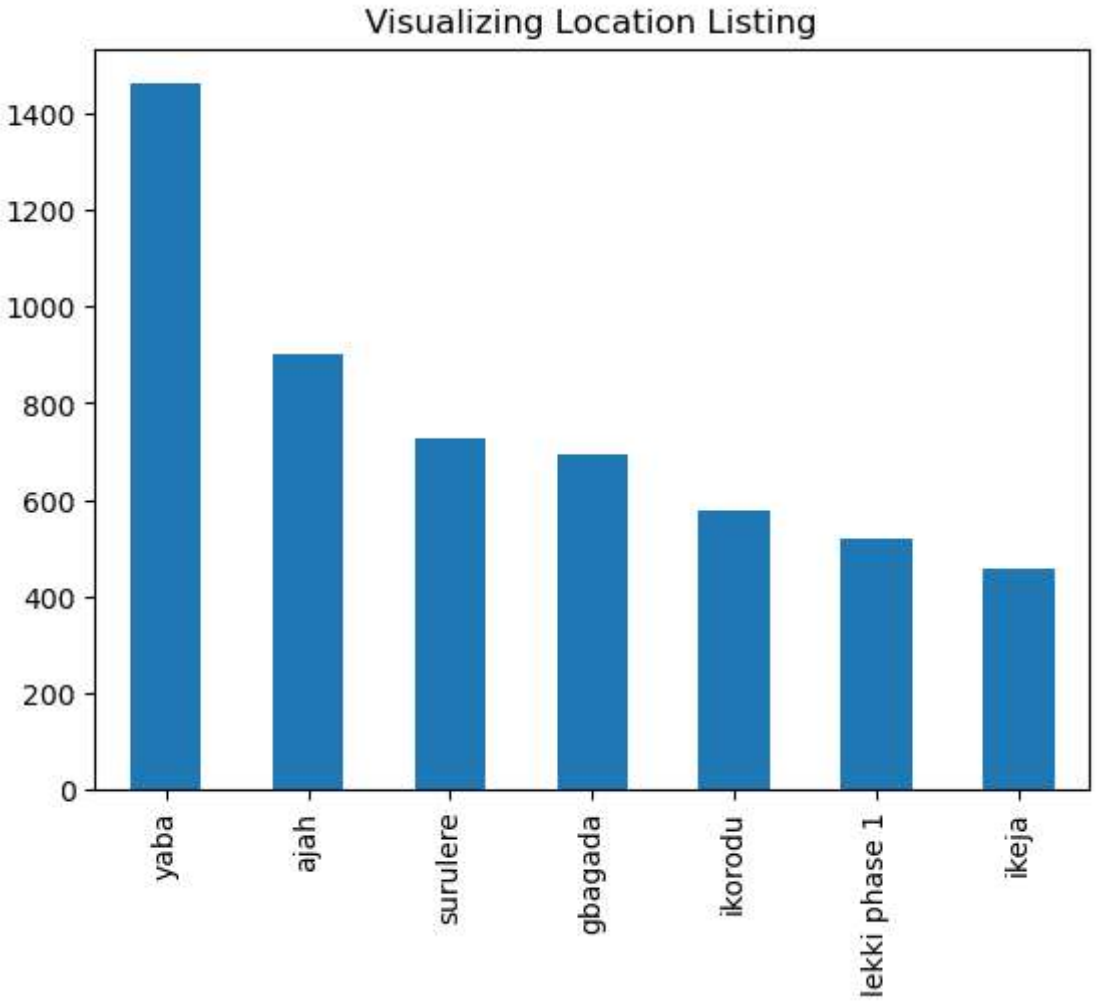
- This means considering one feature or variables

In [14]:
```
# How many listing are there per location?
# First we need to see the listing

count_listing = lagos_houses['location'].value_counts()
count_listing
```

Out[14]:
```
yaba             1460
ajah              900
surulere          728
gbagada           692
ikorodu           578
lekki phase 1     521
ikeja             457
Name: location, dtype: int64
```

In [15]:
```
count_listing.plot.bar()
plt.title('Visualizing Location Listing')
```

Out[15]:
```
Text(0.5, 1.0, 'Visualizing Location Listing')
```

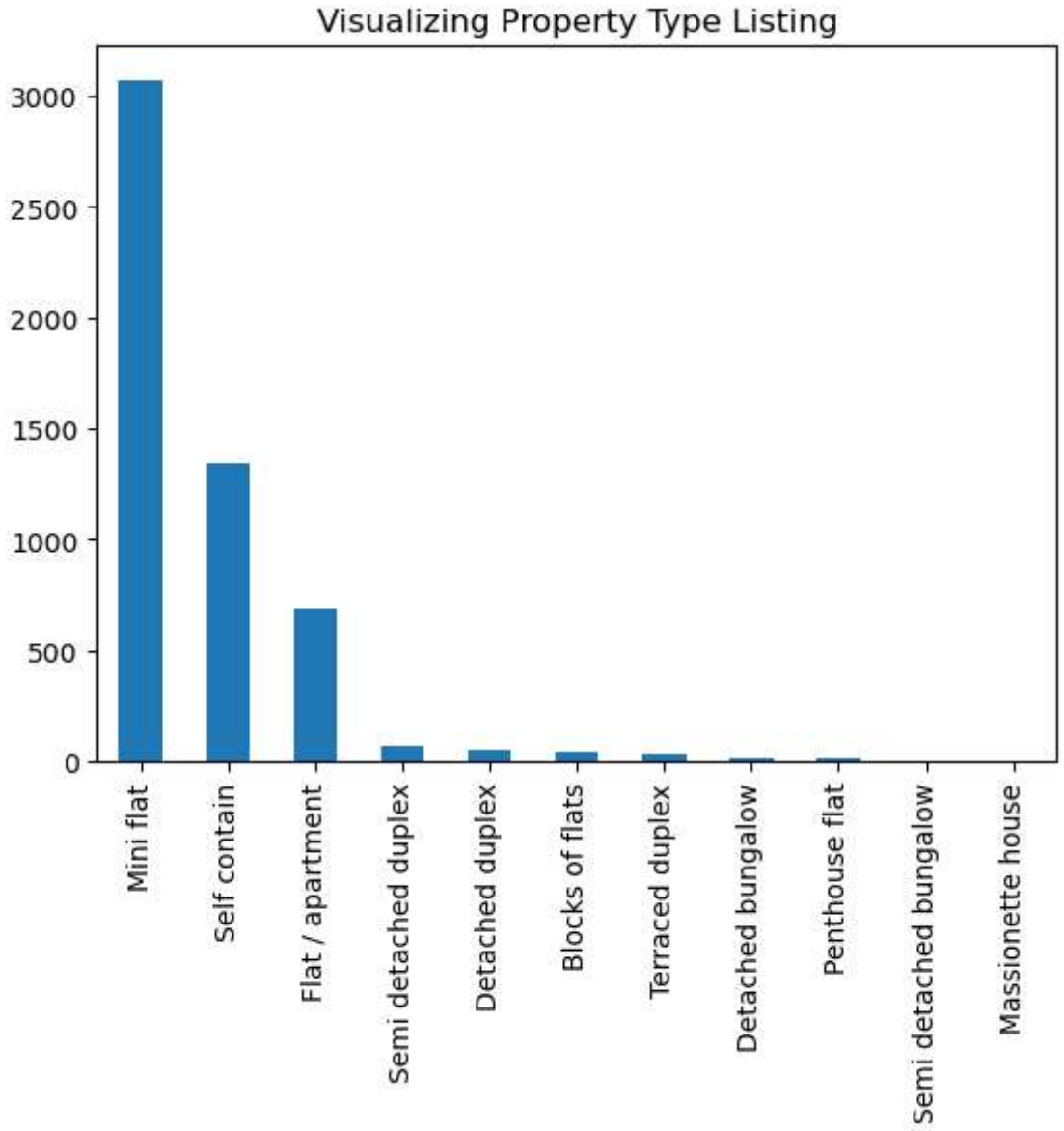


In [16]:
```
count_listing = lagos_houses['Property_Type'].value_counts()
count_listing
```

Out[16]:
```
Mini flat                3070
Self contain             1345
Flat / apartment          686
Semi detached duplex       66
Detached duplex            53
Blocks of flats            42
Terraced duplex            38
Detached bungalow          18
Penthouse flat             14
Semi detached bungalow      3
Massionette house           1
Name: Property_Type, dtype: int64
```

In [17]:
```
count_listing.plot.bar()
plt.title('Visualizing Property Type Listing')
```

Out[17]:
```
Text(0.5, 1.0, 'Visualizing Property Type Listing')
```



Observation

- It is observed that Yaba has the highest listing of about 1460 while Ikeja has the lowest of about 457 compared to other locations.

In [18]:
```
# Summary Statistic per location and Price
```

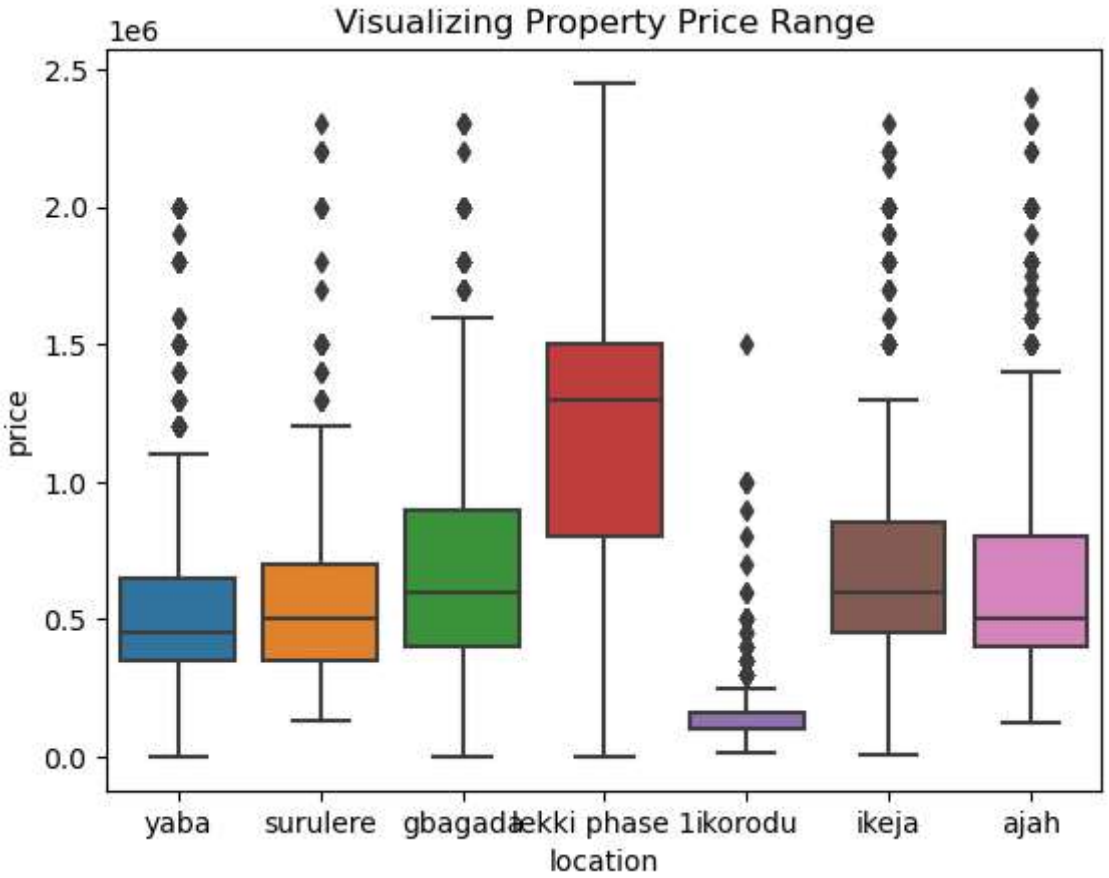
```
lagos_houses.groupby('location').price.describe().astype('int')
```

Out[18]:

location	count	mean	std	min	25%	50%	75%	max
ajah	900	693335	475287	120000	400000	500000	800000	2400000
gbagada	692	742290	451011	150	400000	600000	900000	2300000
ikeja	457	772700	506704	4000	450000	600000	850000	2300000
ikorodu	578	155095	138067	16000	100000	100000	160000	1500000
lekki phase 1	521	1211013	489304	3000	800000	1300000	1500000	2450000
surulere	728	589189	335065	130000	350000	500000	700000	2300000
yaba	1460	550986	331439	250	350000	450000	650000	2000000

In [19]:

```
# We now want to view the distribution of prices by location
sns.boxplot(x = lagos_houses.location, y = lagos_houses.price, data = lagos_houses)
plt.title('Visualizing Property Price Range')
plt.show()
```



Observation

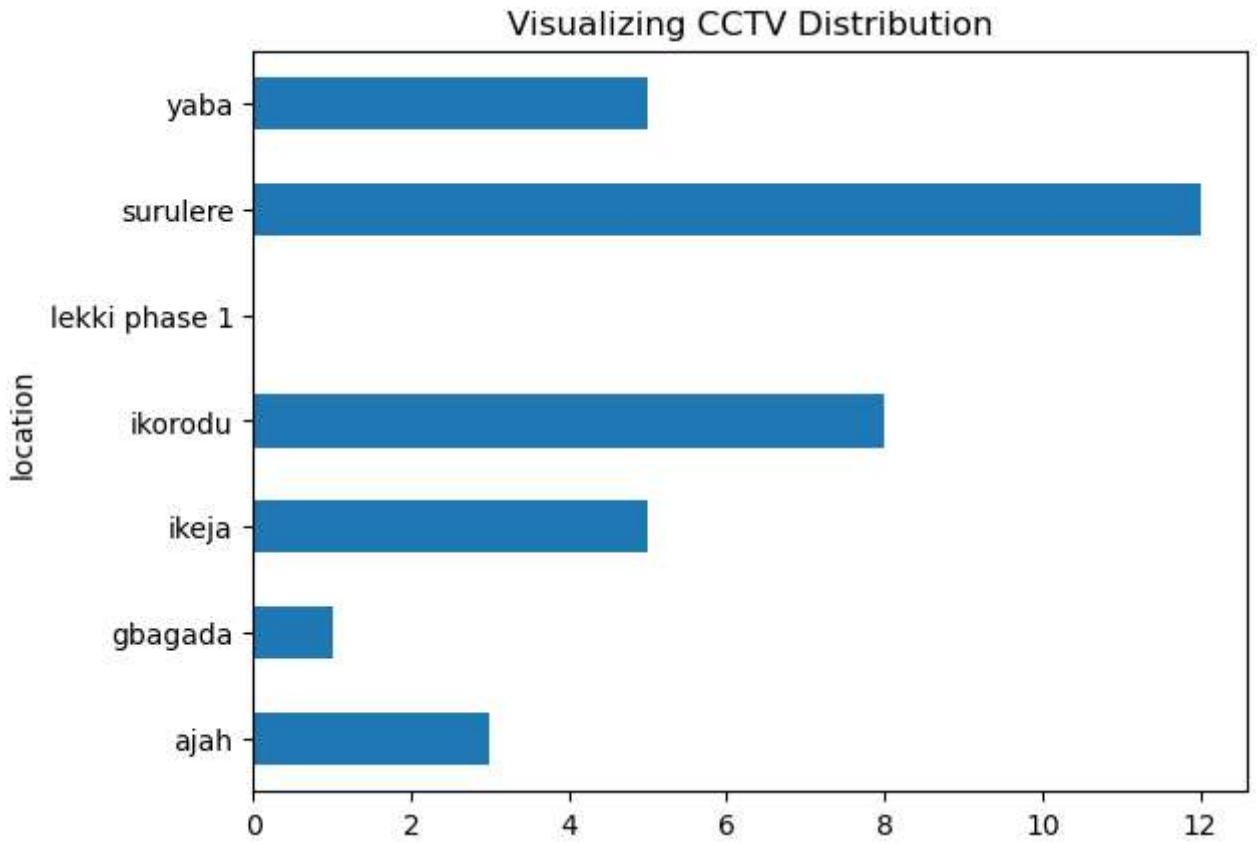
- We can see that the most expensive houses are located in Lekki face one with the median price to be 1.25M, while the list price of houses are in Ikorodu with the median prices to be about 100k compared to other locations

In [20]:
```
# We also want to look at which district is with a good security system like CCTV
cctv_loc = lagos_houses['CCTV'].groupby(lagos_houses.location).sum()
cctv_loc
```

Out[20]:
```
location
ajah              3
gbagada           1
ikeja             5
ikorodu           8
lekki phase 1     0
surulere         12
yaba              5
Name: CCTV, dtype: int64
```

In [21]:
```
# Now we want tp visualize the CCTV Location

cctv_loc.plot.barh()
plt.title('Visualizing CCTV Distribution')
```

Out[21]:
```
Text(0.5, 1.0, 'Visualizing CCTV Distribution')
```

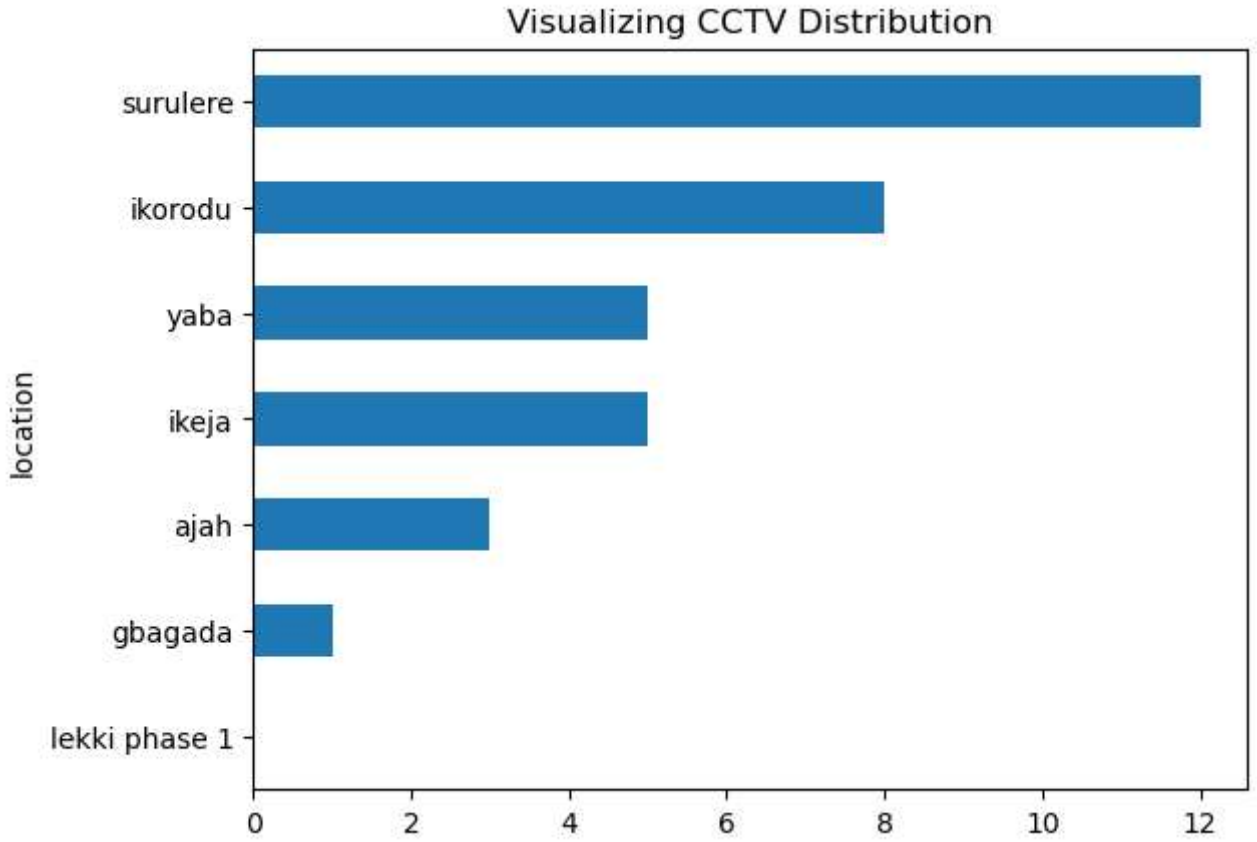


In [22]:
```
# to sort the values from highest to lowest
cctv_loc = lagos_houses['CCTV'].groupby(lagos_houses.location).sum().sort_values()
cctv_loc
```

Out[22]:
```
location
lekki phase 1     0
gbagada           1
ajah              3
ikeja             5
yaba              5
ikorodu           8
surulere         12
Name: CCTV, dtype: int64
```

In [23]:
```
cctv_loc.plot.barh()
plt.title('Visualizing CCTV Distribution')
```

Out[23]:
```
Text(0.5, 1.0, 'Visualizing CCTV Distribution')
```



Observation

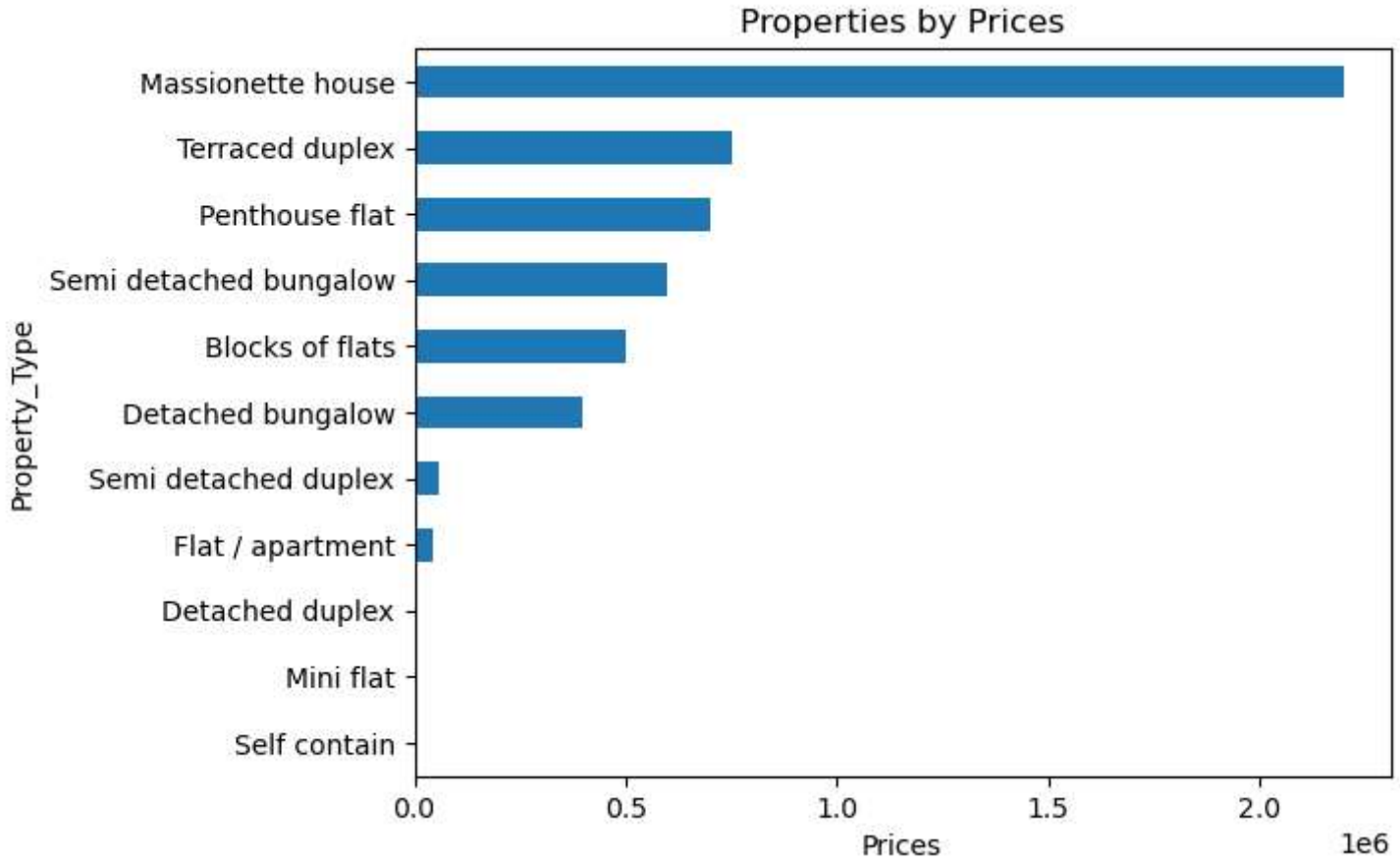
- It is observed that surulere has the Highest number of CCTV cameras of abouts 12 while Lekki Phase1 has none compared to other locations

In [24]:
```
# Houses type that are the cheapest for the highest numbers of room for families that wante an affordable house

min_loc = lagos_houses['price'].groupby(lagos_houses.Property_Type).min().sort_values()
min_loc
```

Out[24]:
```
Property_Type
Self contain                  150.0
Mini flat                    3000.0
Detached duplex              4000.0
Flat / apartment            45000.0
Semi detached duplex        60000.0
Detached bungalow          400000.0
Blocks of flats            500000.0
Semi detached bungalow     600000.0
Penthouse flat             700000.0
Terraced duplex            750000.0
Massionette house         2200000.0
Name: price, dtype: float64
```

In [25]:
```
# Let us now visualize the cheapest property
min_loc.plot.barh()
plt.xlabel('Prices')
plt.title('Properties by Prices')
```

Out[25]:
```
Text(0.5, 1.0, 'Properties by Prices')
```



Observation

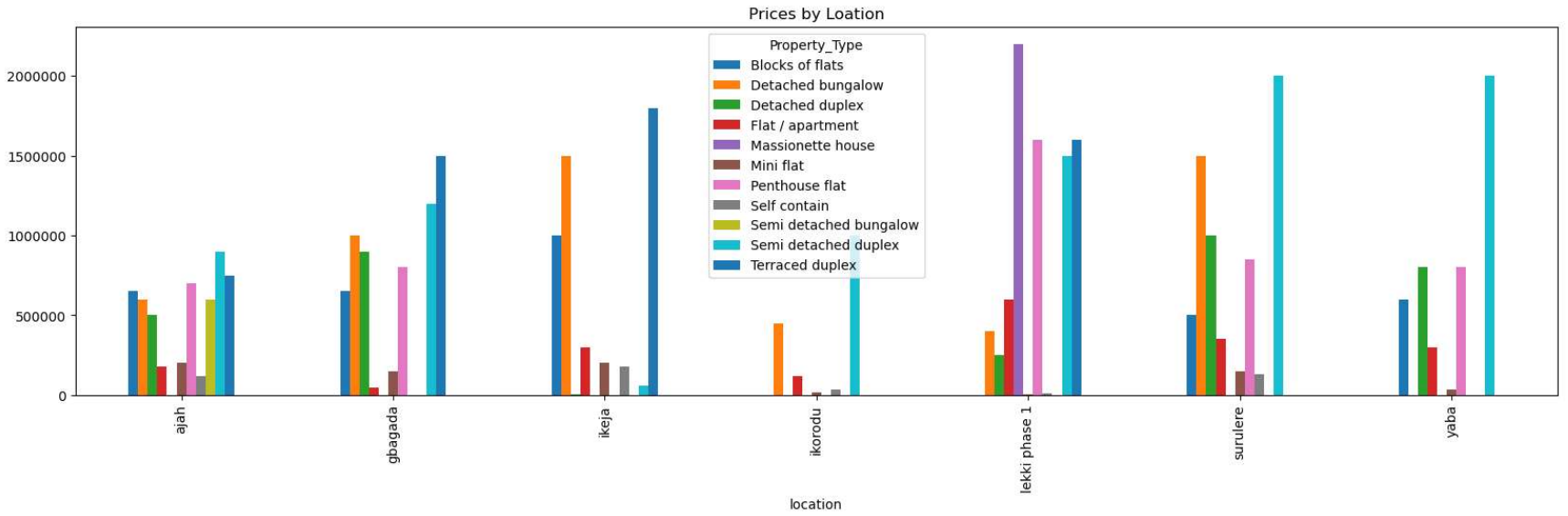
- It is noticed that self contain is the most affordable at the rate of 150 followed by mini flat of about 3000 in price compared to others

In [26]:
```
# We now want to know the location of the cheapest self contain
self_loc = lagos_houses.groupby(['location', 'Property_Type']).price.min().astype('int').unstack('Property_Type')
self_loc
```

Out[26]:

Property_Type	Blocks of flats	Detached bungalow	Detached duplex	Flat / apartment	Massionette house	Mini flat	Penthouse flat	Self contain	Semi detached bungalow	Semi detached duplex	Terraced duplex
location											
ajah	650000.0	600000.0	500000.0	180000.0	NaN	200000.0	700000.0	120000.0	600000.0	900000.0	750000.0
gbagada	650000.0	1000000.0	900000.0	45000.0	NaN	150000.0	800000.0	150.0	NaN	1200000.0	1500000.0
ikeja	1000000.0	1500000.0	4000.0	300000.0	NaN	200000.0	NaN	180000.0	NaN	60000.0	1800000.0
ikorodu	NaN	450000.0	NaN	120000.0	NaN	16000.0	NaN	36000.0	NaN	1000000.0	NaN
lekki phase 1	NaN	400000.0	250000.0	600000.0	2200000.0	3000.0	1600000.0	10000.0	NaN	1500000.0	1600000.0
surulere	500000.0	1500000.0	1000000.0	350000.0	NaN	150000.0	850000.0	130000.0	NaN	2000000.0	NaN
yaba	600000.0	NaN	800000.0	300000.0	NaN	35000.0	800000.0	250.0	NaN	2000000.0	NaN

In [27]:
```
# We now want to plot a pivot table to visualize it
self_loc.plot.bar(figsize = (20, 5))
plt.ticklabel_format(style = 'plain', axis = 'y')
plt.title('Prices by Loation')
plt.show()
```



Observation

- It is observed that gbagada has the lowest or cheapest price of appartment followed by Yaba as against other locations

In [28]:
```
# We waant to look at top 3 location by price using pie chart

top3_loc = lagos_houses['price'].groupby(lagos_houses.location).sum().head(3).astype('int').sort_values()
top3_loc
```

Out[28]:
```
location
ikeja      353124000
gbagada    513665150
ajah       624002009
Name: price, dtype: int32
```

In [29]:
```
# Also we can do buttom 3
buttom3_loc = lagos_houses['price'].groupby(lagos_houses.location).sum().tail(3).astype('int').sort_values()
buttom3_loc
```

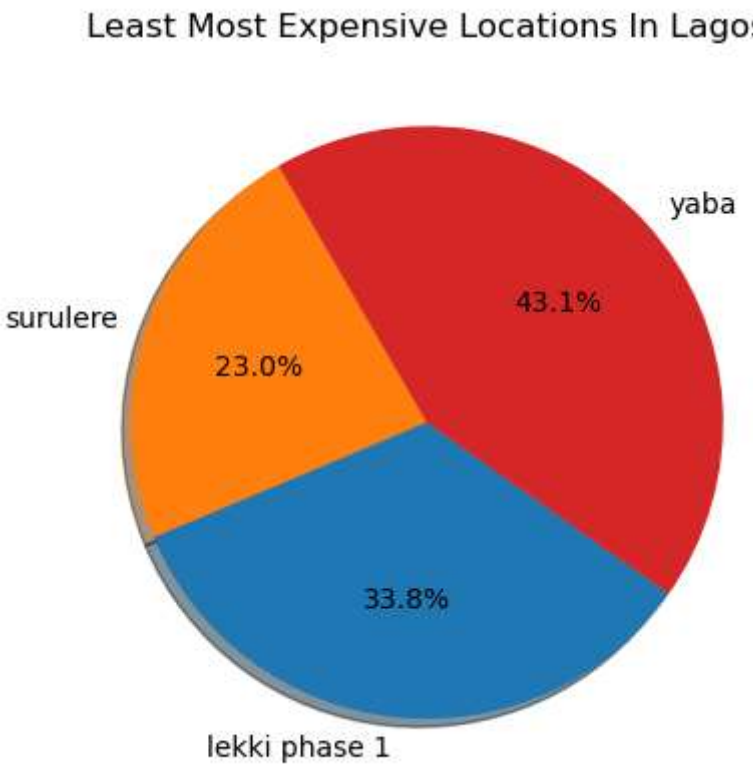
Out[29]:
```
location
surulere         428930009
lekki phase 1    630938000
yaba             804440839
Name: price, dtype: int32
```

In [82]:
```
# We now want to visualize it using pie chart Top 3 Location

colors = [ '#2ca02c', '#ff7f0e', '#d62728', '#Bc564b', '#1f77b4' ]
explode = (0.1, 0, 0, 0, 0)
plt.pie(top3_loc, labels = top3_loc.index, colors = colors, autopct = '%1.1f%%', shadow = True, startangle = 140)
plt.title('Top 3 Most Expensive Locations In Lagos')
plt.show()
```

Obsevation

- It wil be observed that Ajah Lagos has the most expensive rent in lagos with 41.9% high followed by Gbagada 34.5% compared to Ikeja that is 23.7% amongst the top 3 considered.

In [31]:
```
# We now want to visualize it using pie chart Buttom 3 Location

colors = ['#ff7f0e', '#1f77b4', '#d62728', '#2ca02c', '#Bc564b']
explode = (0.1, 0.3, 0, 0, 0)
plt.pie(buttom3_loc, labels = buttom3_loc.index, colors = colors, autopct = '%1.1f%%', shadow = True, startangle = 120)
plt.title('Least Most Expensive Locations In Lagos')
plt.show()
```



Obsevation

- It wil be observed that Surulere Lagos has the least expensive rent in lagos with 23.0% compared to Yaba and Lekki Phase 1.

Bivarite Analysis

- This is used to compare two features.

In [32]:
```
lagos_houses.columns
```

Out[32]:
```
Index(['location', 'bed', 'bath', 'toilet', 'price', 'Property_Type',
       'Parking_Space', 'Security', 'Electricity', 'Furnished',
       'Security_Doors', 'CCTV', 'Pool', 'Gym', 'BQ'],
      dtype='object')
```

In [33]:
```
lagos_houses.head()
```

Out[33]:

	location	bed	bath	toilet	price	Property_Type	Parking_Space	Security	Electricity	Furnished	Security_Doors	CCTV	Pool	Gym	BQ
0	yaba	1	1	2	700000.0	Mini flat	0	0	0	0	0	0	0	0	
1	yaba	1	1	2	700000.0	Mini flat	0	0	0	0	0	0	0	0	
2	yaba	1	1	2	650000.0	Mini flat	0	0	0	0	0	0	0	0	
3	yaba	1	1	1	450000.0	Mini flat	0	0	0	0	0	0	0	0	
4	yaba	3	3	4	800000.0	Detached duplex	0	1	0	0	0	0	0	0	

In [34]:
```
# We now want to visualize using the 2 categorical variable here which is the property type and location
# Looking at the buttom 10 locations by price

lagos_houses.tail(3)
```

Out[34]:

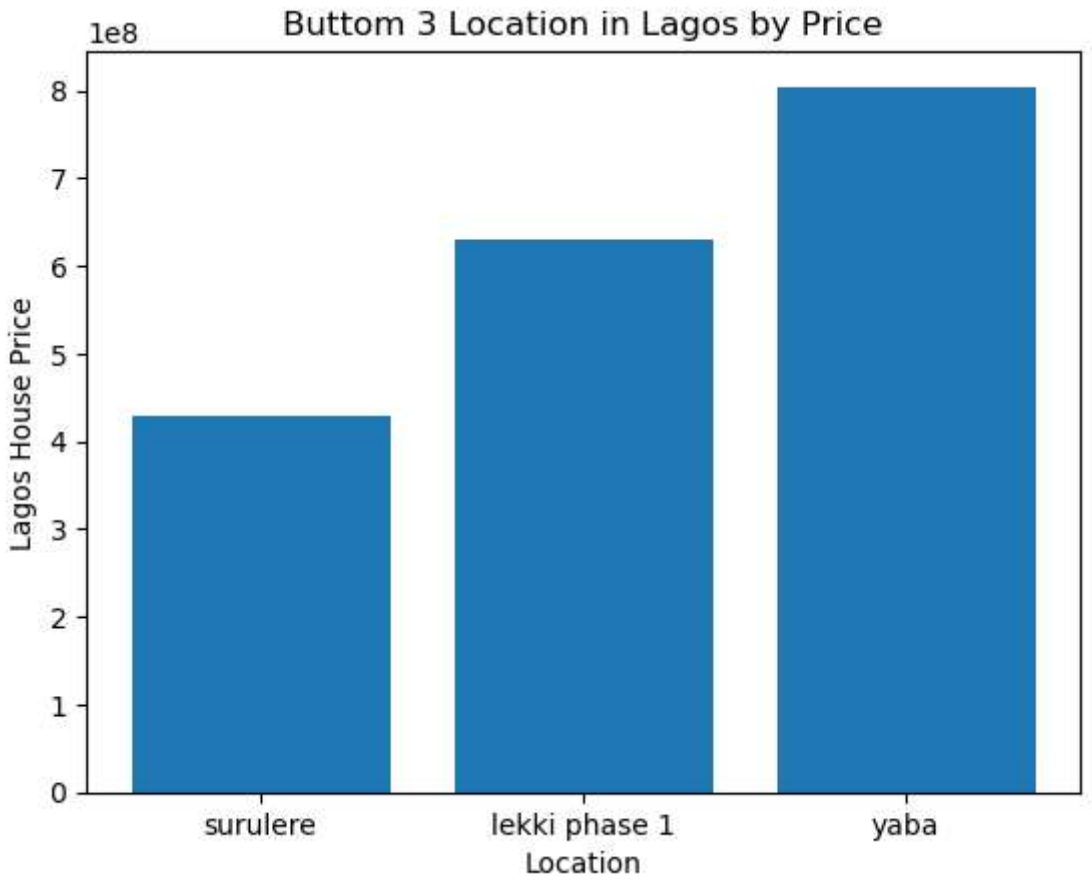
	location	bed	bath	toilet	price	Property_Type	Parking_Space	Security	Electricity	Furnished	Security_Doors	CCTV	Pool	Gyn
5333	ajah	4	4	5	1700000.0	Semi detached duplex	1	0	0	0	0	0	0	
5334	ajah	1	1	2	500000.0	Mini flat	0	0	0	0	0	0	0	
5335	ajah	4	4	5	1800000.0	Semi detached duplex	1	1	0	1	0	0	0	

In [5]:
```
bot_loc = lagos_houses.groupby(['location']).price.sum().tail(3).astype('int').sort_values()
bot_loc
```

Out[5]:
```
location
surulere         428930009
lekki phase 1    630938000
yaba             804440839
Name: price, dtype: int32
```

In [6]:
```
# We want to visualie using the Bar Chart
plt.bar(x = bot_loc.index, height = bot_loc)

plt.title('Buttom 3 Location in Lagos by Price')
plt.ylabel('Lagos House Price')
```

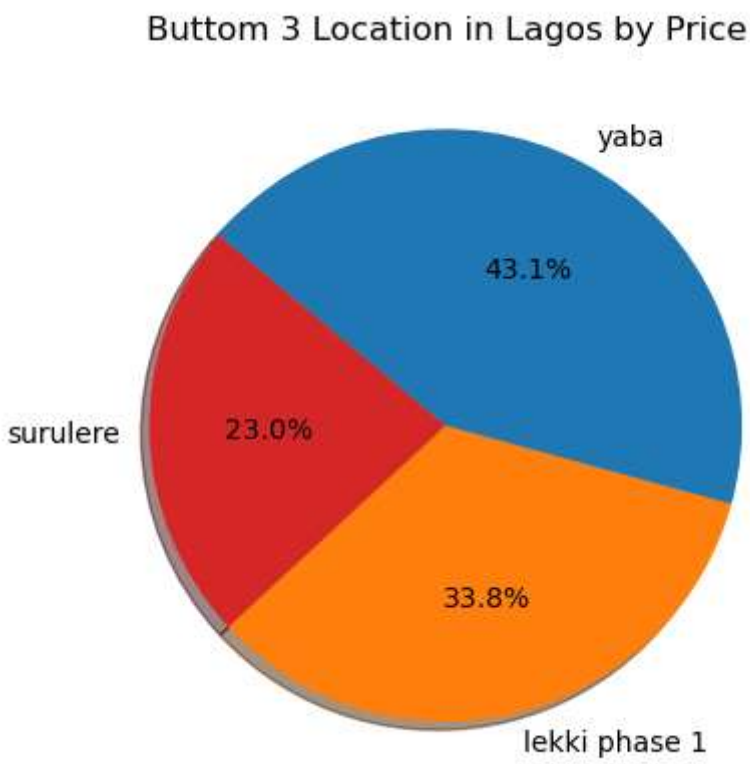
```
plt.xlabel('Location')
plt.show()
```



Observation

- Using the Bar chart we comapred two features Price of rent and Location, Surulere has the cheapest rent compared to Lekki Phase1 and Yaba as we look into the buttom 3

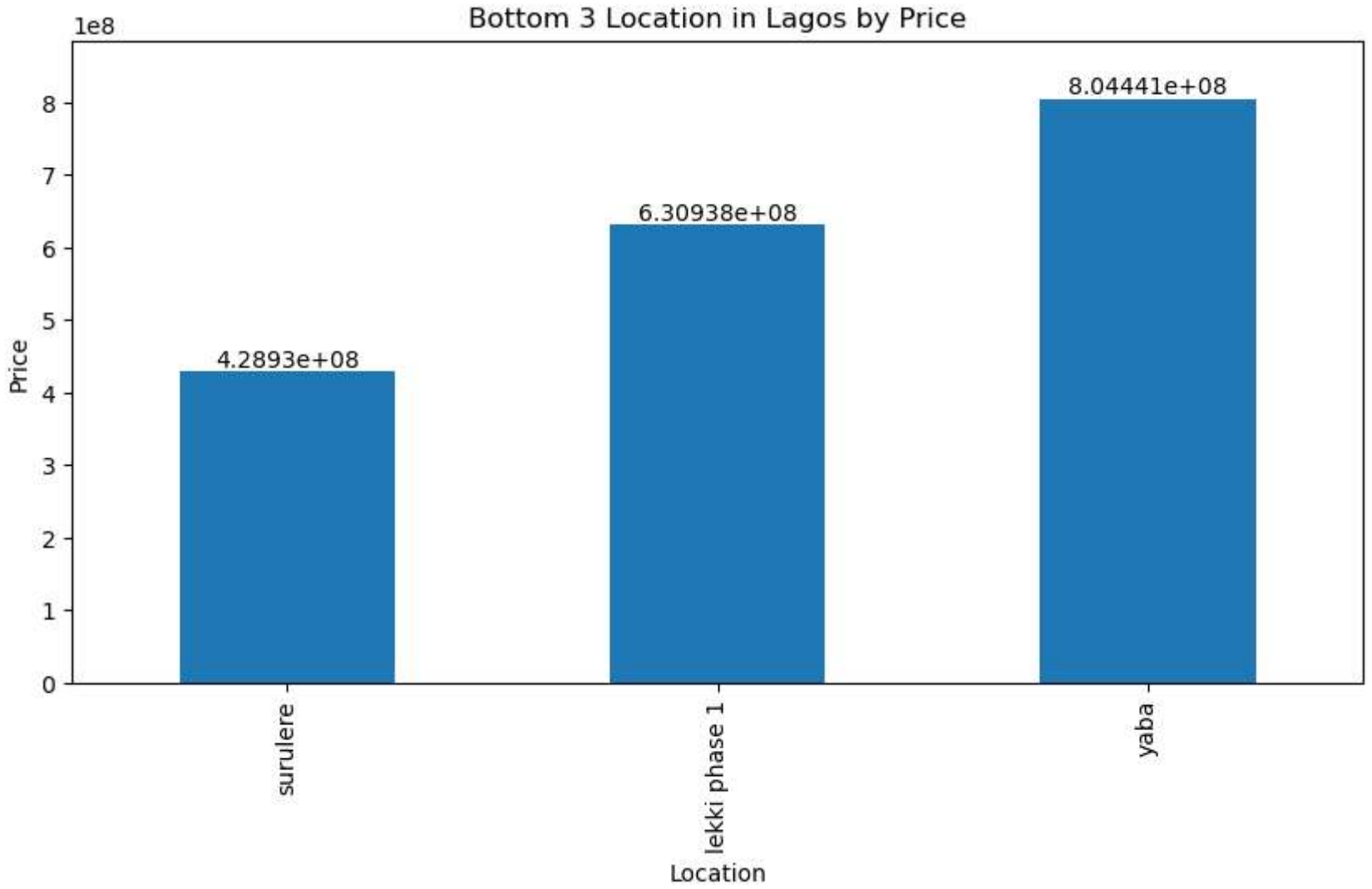
In [37]:
```
# Using pie chart for the Buttom 3 Locations in Lagos as above

colors = ['#d62728', '#ff7f0e', '#1f77b4', '#2ca02c', '#Bc564b']
explode = (0.1, 0.3, 0, 0, 0.1)
plt.pie(bot_loc, labels = bot_loc.index, colors = colors, autopct = '%1.1f%%', shadow = True, startangle = 140)
plt.title('Buttom 3 Location in Lagos by Price')
plt.show()
```



Observation

- Using the pie chart we still comapred two features Price of rent and Location, Surulere has cheapest rent 23.0% compared to Lekki Phase1 and Yaba in the buttom 3

In [38]:
```
# We want to visualize using the Bar Chart and show labels

ax = bot_loc.plot(kind = 'bar', figsize = (10, 5), title = 'Bottom 3 Location in Lagos by Price', xlabel = 'Location',
                  ylabel = 'Price', legend = False)

# Anotate
ax.bar_label(ax.containers[0], label_type = 'edge')

# we want to pad the spacing between the number and the edge of the figure
ax.margins(y = 0.1)
```



Observation

- With the prices of the rent location on the bars we could see the likely prices in these areas. Surulere is still the most affordable area in terms off price. followed by Lekki Phase1 and then Yaba

In [28]:
```
# We now want to plot a doughnut chart

colors = ['#d62728', '#ff7f0e', '#1f77b4', '#2ca02c', '#Bc564b']
explode = (0.0, 0.0, 0.0)
```

```
#Doughnut chart
plt.pie(bot_loc, labels = bot_loc.index, colors = colors, autopct = '%1.1f%%', pctdistance = 0.85, explode = explode)

# Then we will draw a circle for the doughnut chart
centre_circle = plt.Circle((0, 0), 0.7, fc = 'white')
fig = plt.gcf()

# Adding circle in pie chart
fig.gca().add_artist(centre_circle)

#Add Legend
plt.legend(bot_loc.index, loc = 'upper left')

plt.title('Buttom 3 Location in Lagos by Price')
plt.show()
```

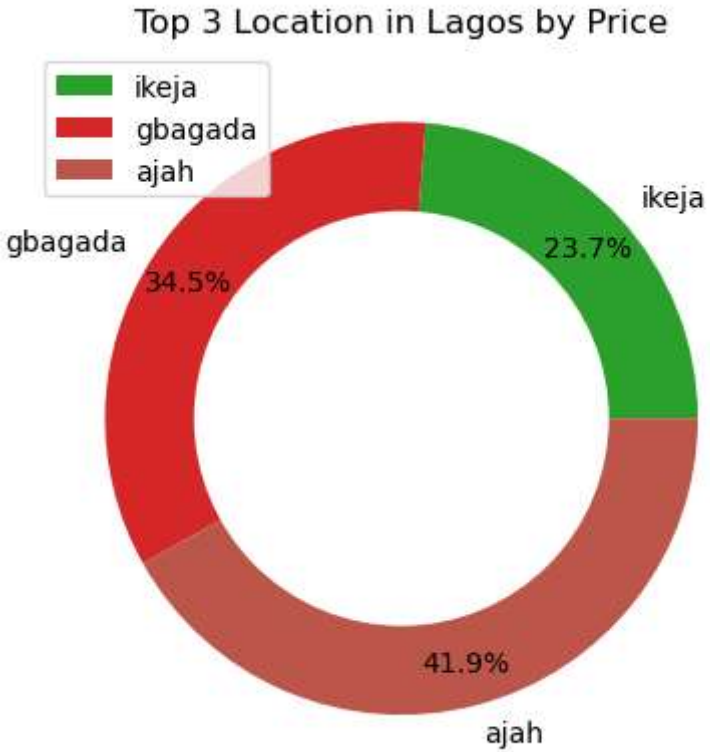
Observation

- As seen in the above charts this is another visual on the bottom 3 using the doughnut chart

In [30]:
```
top3_loc = lagos_houses['price'].groupby(lagos_houses.location).sum().head(3).astype('int').sort_values()
top3_loc
```

Out[30]:
```
location
ikeja      353124000
gbagada    513665150
ajah       624002009
Name: price, dtype: int32
```

In [31]:
```
# We want to look at top 3 location and price using doughnut chart
colors = ['#2ca02c', '#d62728', '#Bc564b', '#ff7f0e', '#1f77b4']
explode = (0.0, 0.0, 0.0)

#Doughnut chart
plt.pie(top3_loc, labels = top3_loc.index, colors = colors, autopct = '%1.1f%%', pctdistance = 0.85, explode = explode)

# Then we will draw a circle for the doughnut chart
centre_circle = plt.Circle((0, 0), 0.70, fc = 'white')
fig = plt.gcf()

# Adding circle in pie chart
fig.gca().add_artist(centre_circle)

#Add Legend
plt.legend(top3_loc.index, loc = 'upper left')

plt.title('Top 3 Location in Lagos by Price')
plt.show()
```



Observation

- In this doughnut chart we have Ajah 41.9% as the most expensive followed Gbagada and Ikeja.

In [41]:
```
# Location with Swiming Pool Facilities
pool_loc = lagos_houses['Pool'].groupby(lagos_houses.location).value_counts().sort_values()
pool_loc
```

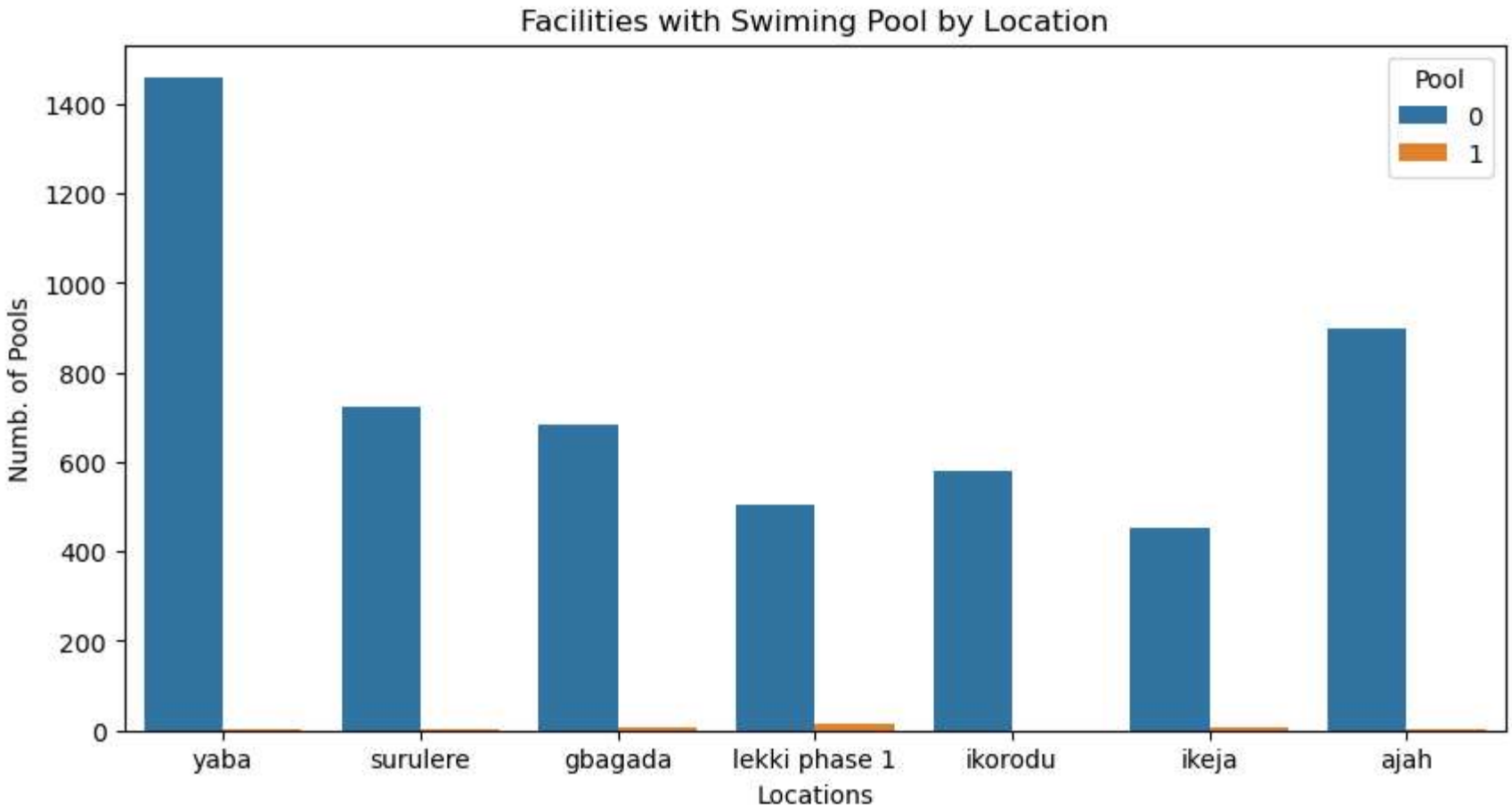
Out[41]:

location	Pool	
ajah	1	1
yaba	1	1
surulere	1	3
ikeja	1	5
gbagada	1	7
lekki phase 1	1	15
ikeja	0	452
lekki phase 1	0	506
ikorodu	0	578
gbagada	0	685
surulere	0	725
ajah	0	899
yaba	0	1459

Name: Pool, dtype: int64

In [42]:

```
# We want to create visualization for the data above
plt.figure(figsize = (10, 5))
plt.title('Facilities with Swiming Pool by Location')
sns.countplot(x = 'location', data = lagos_houses, hue = 'Pool')
plt.xlabel('Locations')
plt.ylabel('Numb. of Pools')
plt.show()
```



Observation

- In this chart we notice that Lekki Phase1 Has the hiest number of appartments with swimming pool, followed by Gbagada, Ikeja and others as represented in the yelow bars. Meanwhile the blue bars are the appartments without swiming pools

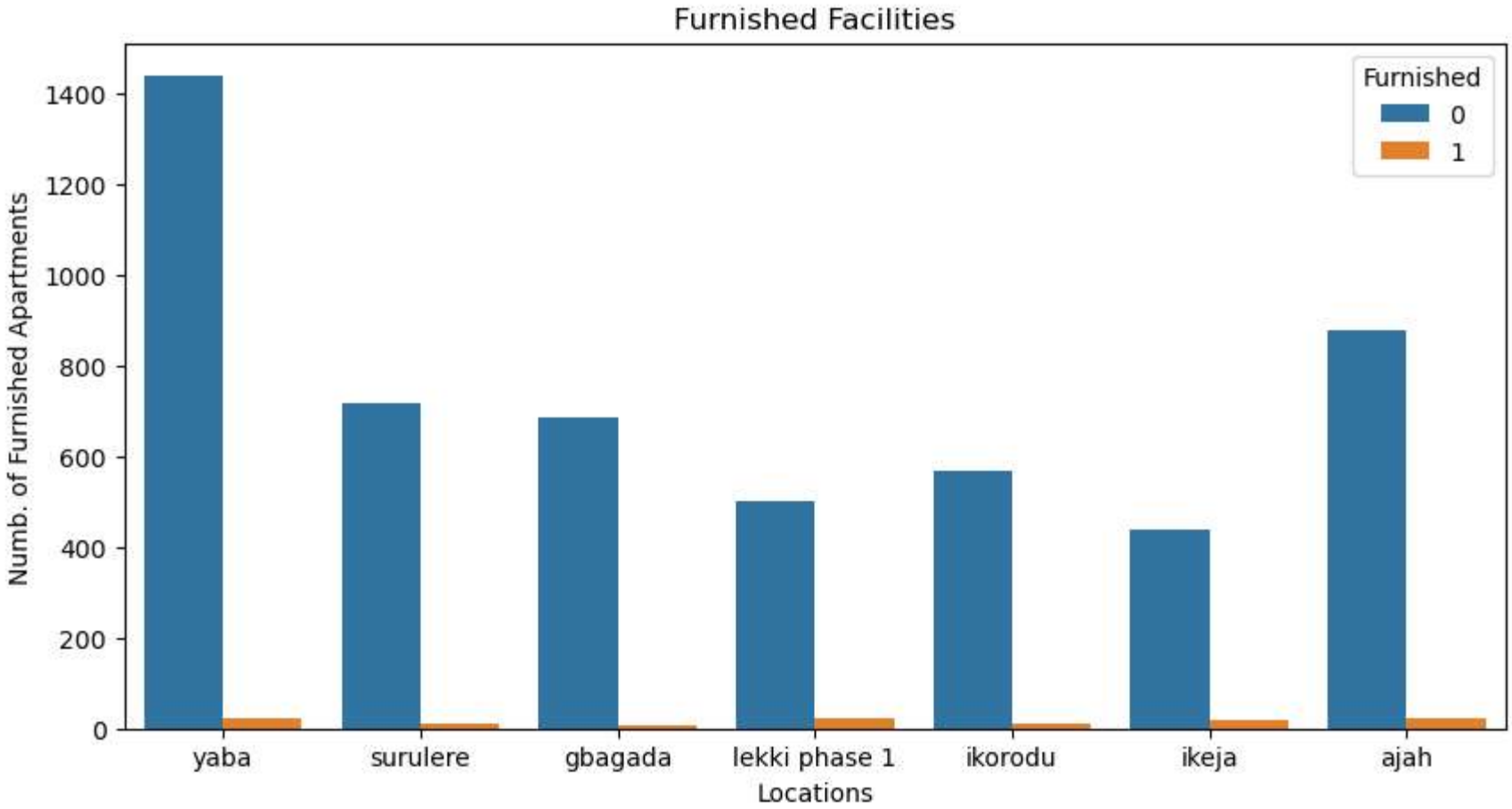
In [43]:
```
# Location with Furnished Apartments

fur_loc = lagos_houses['Furnished'].groupby(lagos_houses.location).value_counts().sort_values()
fur_loc
```

Out[43]:

location	Furnished	
gbagada	1	5
ikorodu	1	9
surulere	1	12
ikeja	1	17
lekki phase 1	1	21
yaba	1	21
ajah	1	23
ikeja	0	440
lekki phase 1	0	500
ikorodu	0	569
gbagada	0	687
surulere	0	716
ajah	0	877
yaba	0	1439

Name: Furnished, dtype: int64

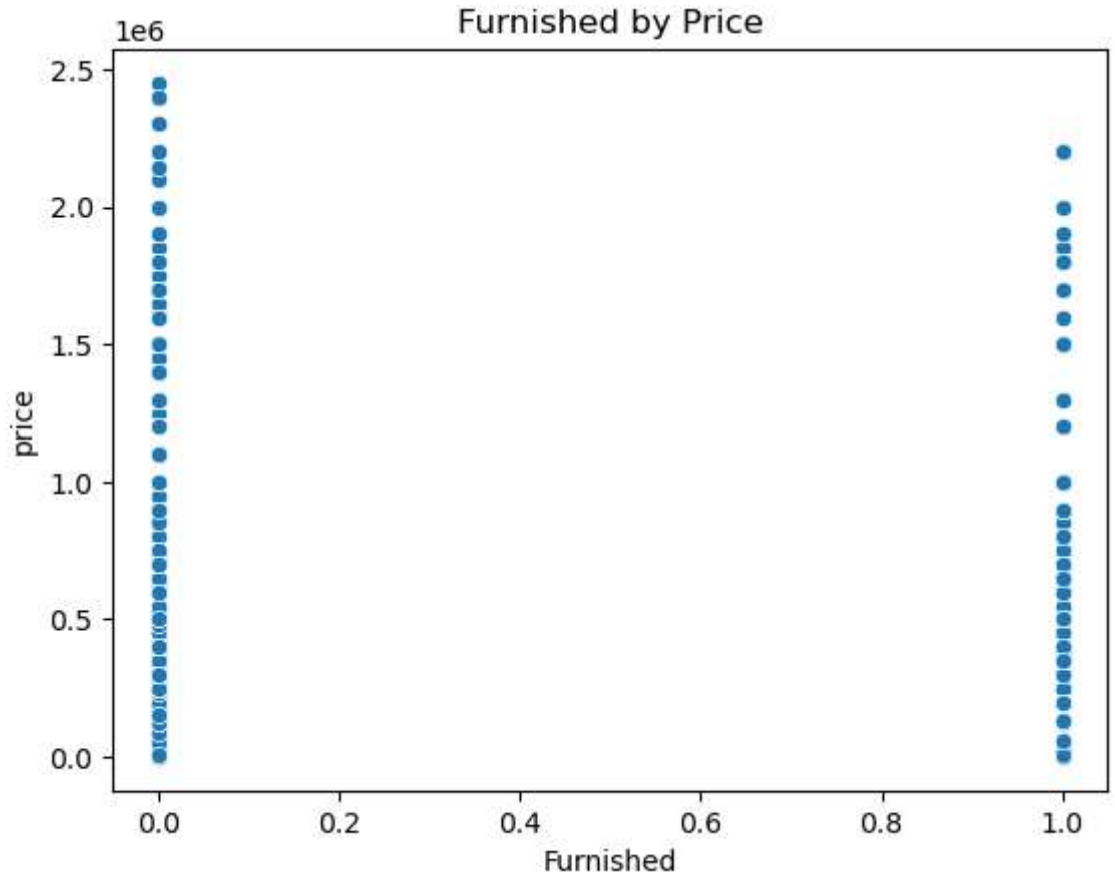
In [44]:
```
plt.figure(figsize = (10, 5))
plt.title('Furnished Facilities')
sns.countplot(x = 'location', data = lagos_houses, hue = 'Furnished')
plt.xlabel('Locations')
plt.ylabel('Numb. of Furnished Apartments')
plt.show()
```



Observation

- Here appartmens that are furnished only in few places. Location with highest furnished appartments are Ajah, Yaba, Lekki Phase 1, then Ikeja and others as seen in Yellow Bar.

In [45]:
```
# Now we want to compare 2 numerical valuees using scattered polts as it is used to compare relationship between two qu

sns.scatterplot(x = lagos_houses['Furnished'], y = lagos_houses['price'])
plt.title('Furnished by Price')
plt.show()
```



Observation

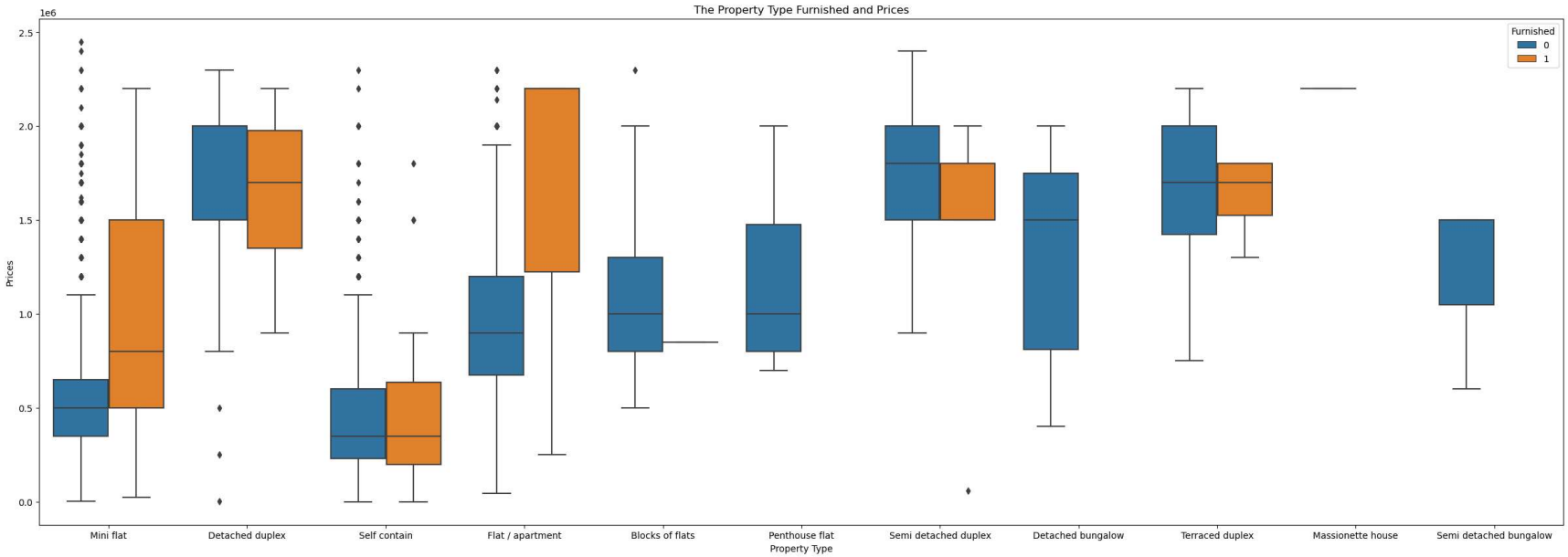
- Using Scattered plot, we were able to get insight into the Furnished appartments and the prices. The clusstered region shows the numbers furnished appartment the cost as we see to the right side of the chat. The furnished appartments seem limited in number compared to the one that is not.

Multivariate Analysis

- Comparing more than two features or variables

In [46]:
```
# Visualizing Property Type, Furnished by Price using box plot

plt.figure(figsize = (30, 10))
```

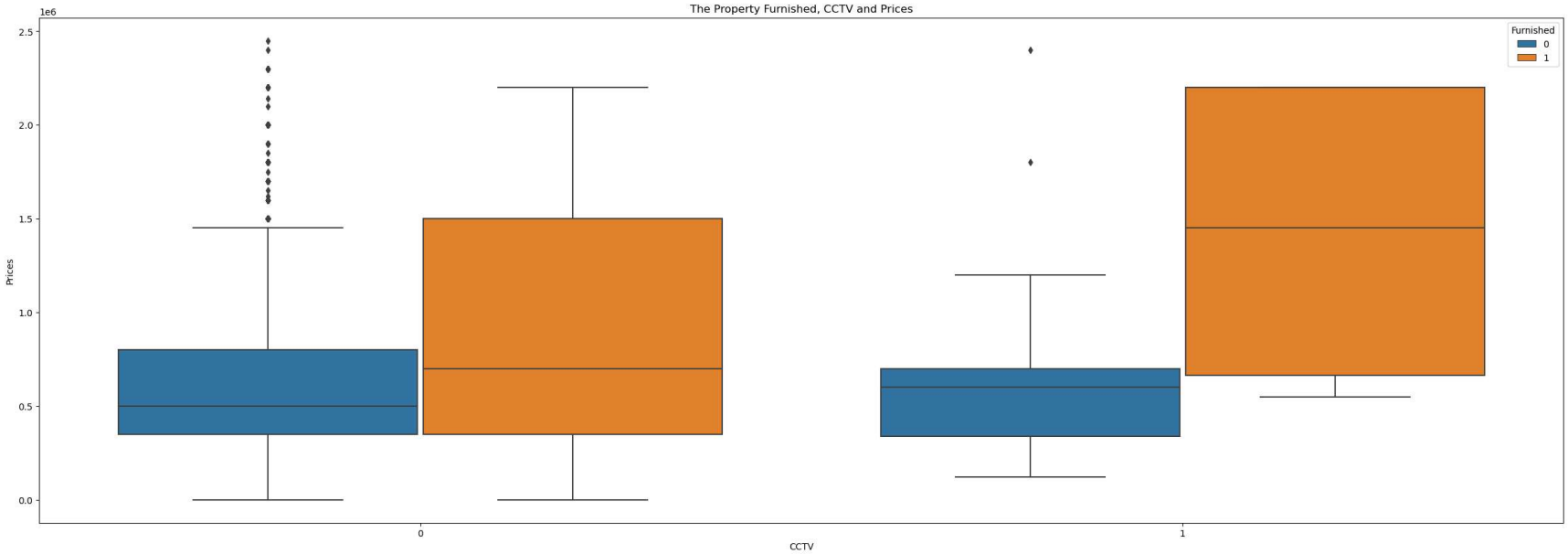
```
sns.boxplot(x = 'Property_Type', y = 'price', data = lagos_houses, hue = 'Furnished')
plt.title('The Property Type Furnished and Prices')
plt.xlabel('Property Type')
plt.ylabel('Prices')
plt.show()
```



Observation

- Here we will observe that the miniflats are the types of furnished appartments that are affordable, while self contain is most affordable and flats has the highest range in price and avalability.

In [47]:
```
# We now want to visalize the houses Furnished, with CCTV and the Price

plt.figure(figsize = (30, 10))
sns.boxplot(x = 'CCTV', y = 'price', data = lagos_houses, hue = 'Furnished')
plt.title('The Property Furnished, CCTV and Prices')
plt.xlabel('CCTV')
plt.ylabel('Prices')
plt.show()
```



In [48]:
```
# Using Lm Chart

plt.figure(figsize = (30, 10))
sns.lmplot(x = 'CCTV', y = 'price', data = lagos_houses, hue = 'Furnished')
plt.title('The Property Furnished, CCTV and Prices')
plt.xlabel('CCTV')
plt.ylabel('Prices')
plt.show()
```

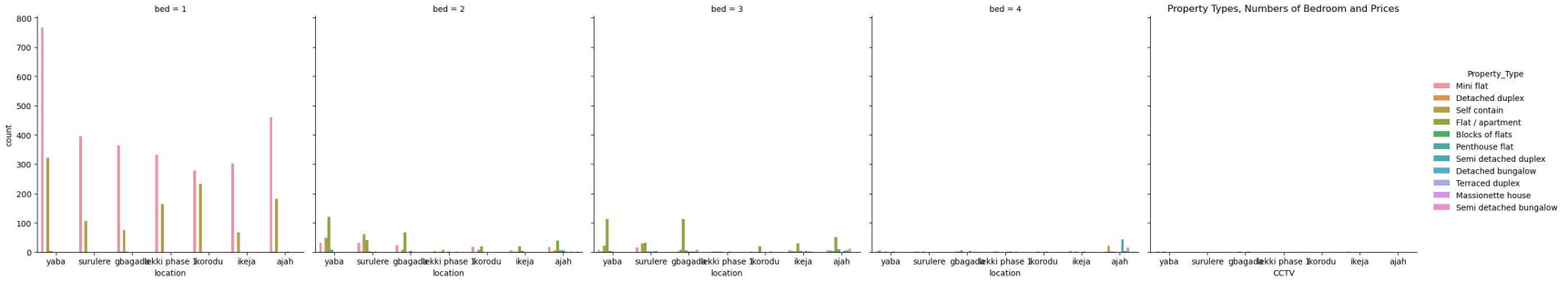
```
<Figure size 3000x1000 with 0 Axes>
```

Observation

- We will observe that most Furnished appartments does not fal into areas with CCTV cameras

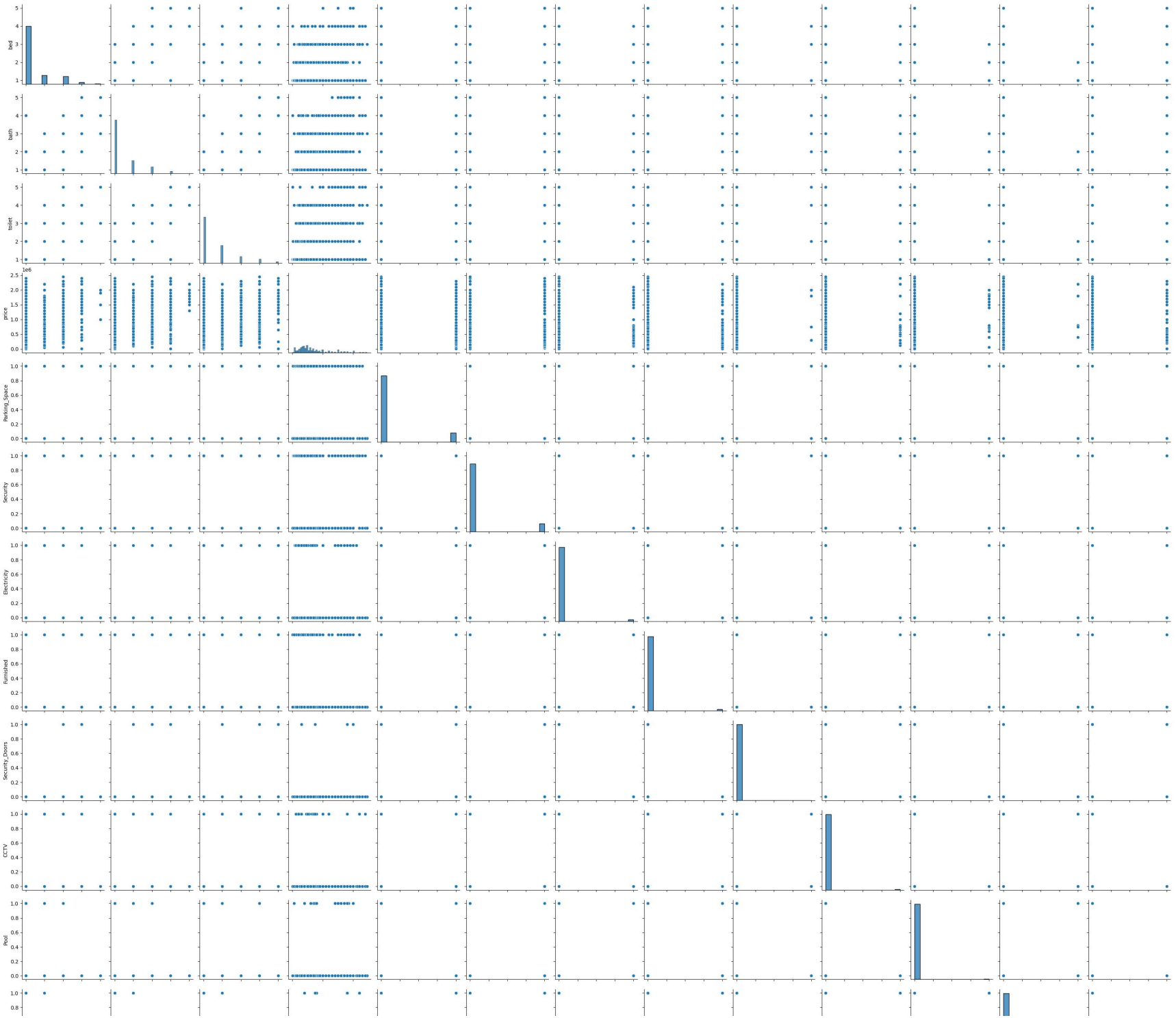
In [49]:
```
#catplot visualizing the number of bedrooms and property type by location.

plt.figure(figsize = (20, 10))
sns.catplot(col = 'bed', x = 'location', kind = 'count', data = lagos_houses, hue = 'Property_Type')
plt.title('Property Types, Numbers of Bedroom and Prices')
plt.xlabel('CCTV')
plt.ylabel('Prices')
plt.show()
```

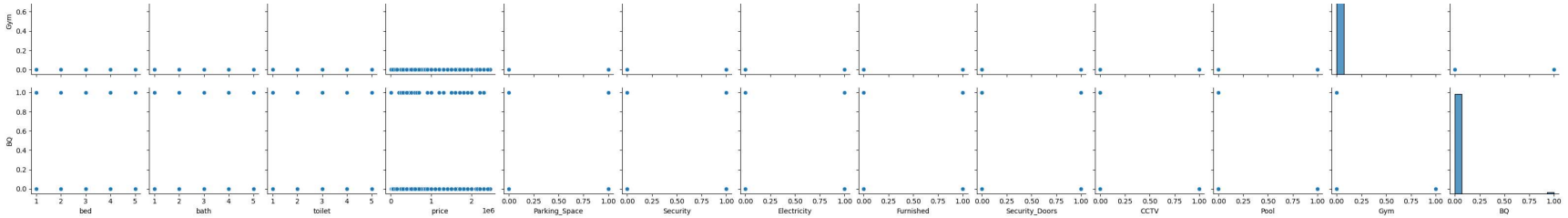
```
<Figure size 2000x1000 with 0 Axes>
```



In [50]:
```
#Pair plot captures the relationship between two features. It captures the entire columns
sns.pairplot(data = lagos_houses)
```

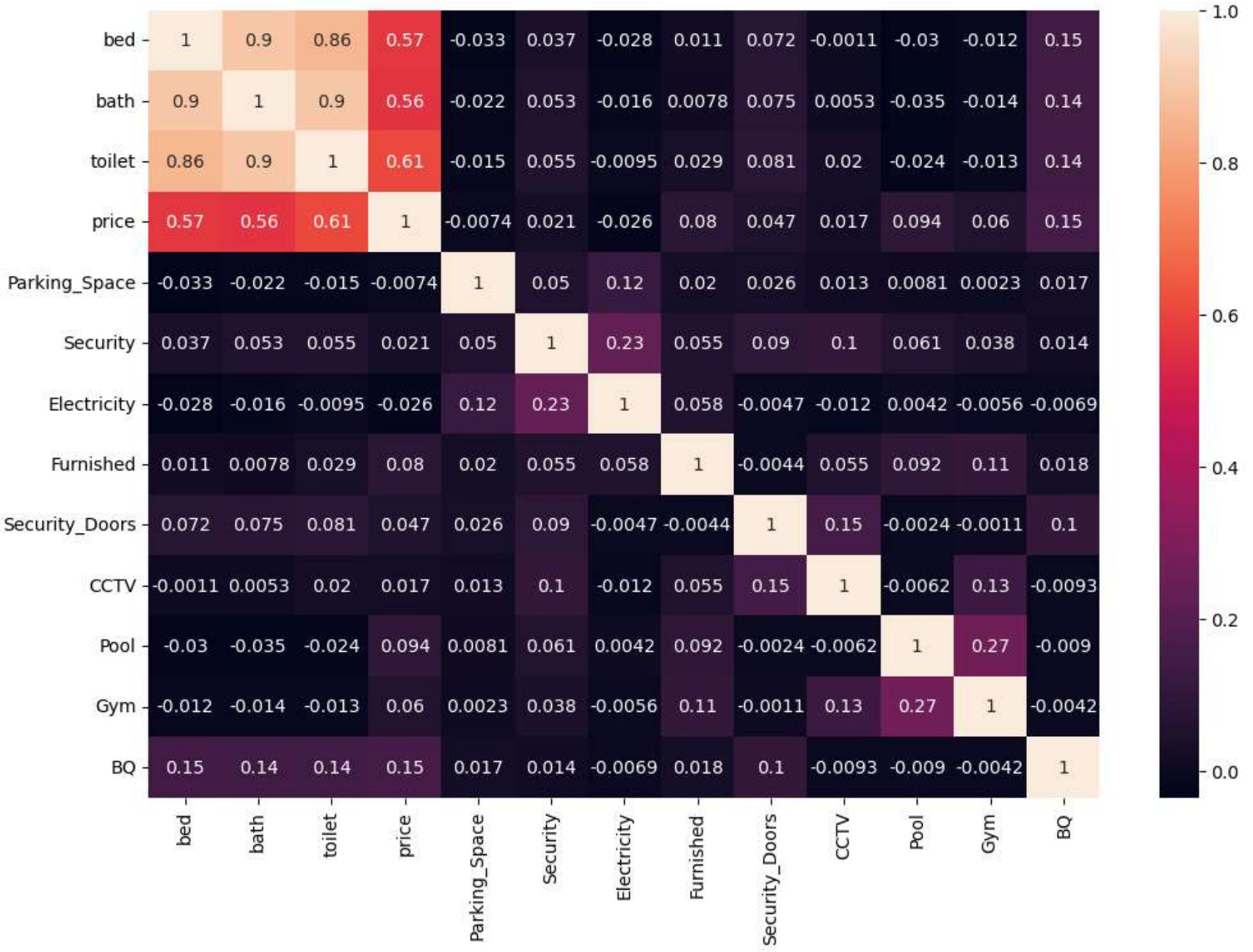
Out[50]:
```
<seaborn.axisgrid.PairGrid at 0x29801430610>
```


bed
bath
toilet
price
1e6
Parking_Space
Security
Electricity
Furnished
Security_Doors
CCTV
Pool



In [51]:
```
# Using correlation we will be using Hitmap

plt.figure(figsize = (12, 8))
corel = lagos_houses.corr()
sns.heatmap(corel, annot = True)
```

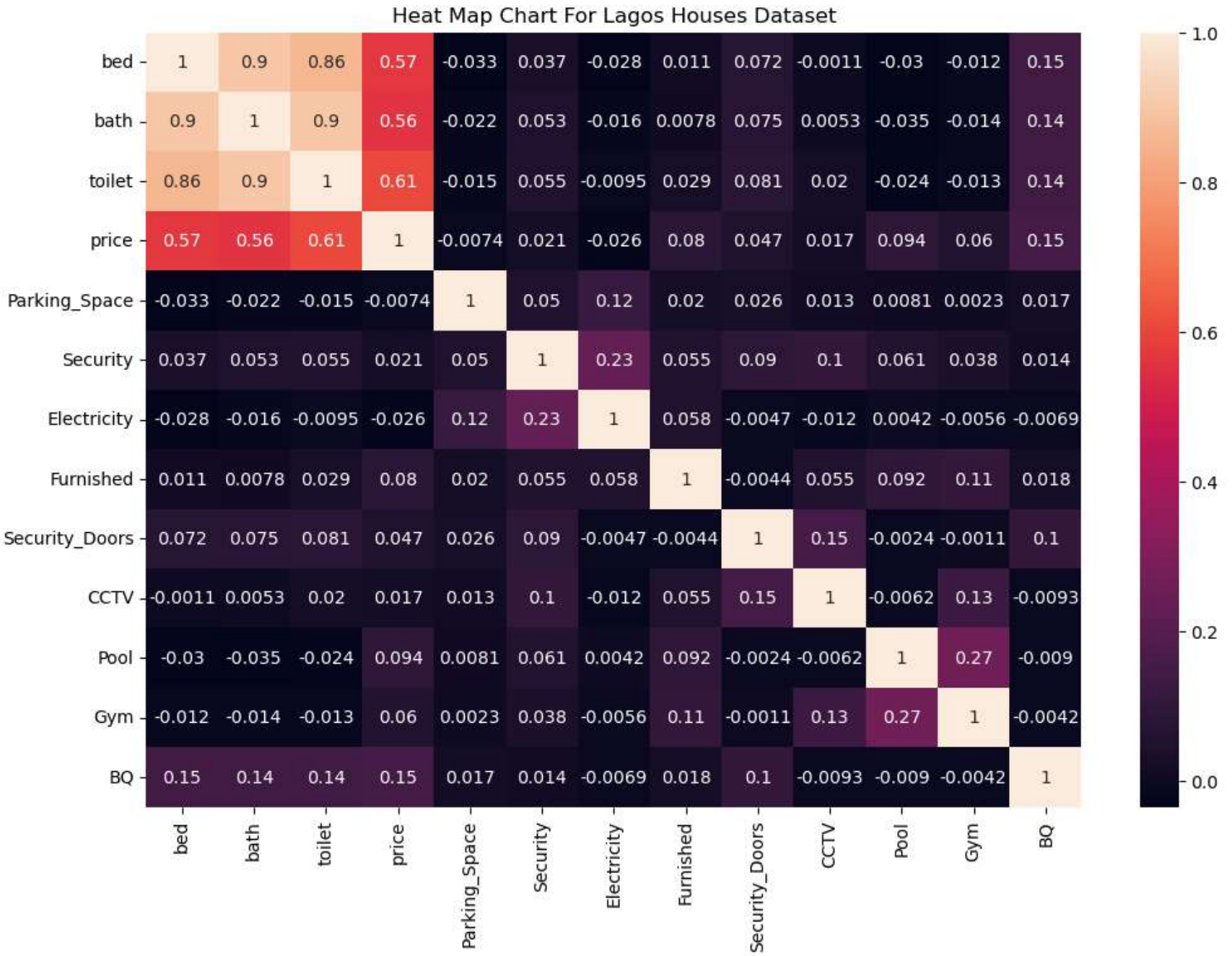
Out[51]:
```
<AxesSubplot:>
```



In [52]:
```
plt.figure(figsize = (12, 8))
corel = lagos_houses.corr()
```

```
sns.heatmap(corel, annot = True)
plt.title('Heat Map Chart For Lagos Houses Dataset')
```

Out[52]:
```
Text(0.5, 1.0, 'Heat Map Chart For Lagos Houses Dataset')
```


Heat Map Chart For Lagos Houses Dataset
bed 1 0.9 0.86 0.57 -0.033 0.037 -0.028 0.011 0.072 -0.0011 -0.03 -0.012 0.15
bath 0.9 1 0.9 0.56 -0.022 0.053 -0.016 0.0078 0.075 0.0053 -0.035 -0.014 0.14
toilet 0.86 0.9 1 0.61 -0.015 0.055 -0.0095 0.029 0.081 0.02 -0.024 -0.013 0.14
price 0.57 0.56 0.61 1 -0.0074 0.021 -0.026 0.08 0.047 0.017 0.094 0.06 0.15
Parking_Space -0.033 -0.022 -0.015 -0.0074 1 0.05 0.12 0.02 0.026 0.013 0.0081 0.0023 0.017
Security 0.037 0.053 0.055 0.021 0.05 1 0.23 0.055 0.09 0.1 0.061 0.038 0.014
Electricity -0.028 -0.016 -0.0095 -0.026 0.12 0.23 1 0.058 -0.0047 -0.012 0.0042 -0.0056 -0.0069
Furnished 0.011 0.0078 0.029 0.08 0.02 0.055 0.058 1 -0.0044 0.055 0.092 0.11 0.018
Security_Doors 0.072 0.075 0.081 0.047 0.026 0.09 -0.0047 -0.0044 1 0.15 -0.0024 -0.0011 0.1
CCTV -0.0011 0.0053 0.02 0.017 0.013 0.1 -0.012 0.055 0.15 1 -0.0062 0.13 -0.0093
Pool -0.03 -0.035 -0.024 0.094 0.0081 0.061 0.0042 0.092 -0.0024 -0.0062 1 0.27 -0.009
Gym -0.012 -0.014 -0.013 0.06 0.0023 0.038 -0.0056 0.11 -0.0011 0.13 0.27 1 -0.0042
BQ 0.15 0.14 0.14 0.15 0.017 0.014 -0.0069 0.018 0.1 -0.0093 -0.009 -0.0042 1
bed bath toilet price Parking_Space Security Electricity Furnished Security_Doors CCTV Pool Gym BQ
1.0 0.8 0.6 0.4 0.2 0.0

In [ ]:

MACHINE LEARNING

We want to use this dataset LAGOS HOUSES

Predicting Lagos House Prices

- This is a regression task Here we will be lookig at some algorithms as in the following:
- Algrithms are set of predefined codes that are preinstalled in Python for building of model EG:
  - Decision Tree or Decision Classifier
  - Random Forest
  - Linear Regression

However we have to install a Python Library called SK Learning First we will import neccesary Libraries after we take a look at the data set

In [33]:
```
lagos_houses.head()
```

Out[33]:

	location	bed	bath	toilet	price	Property_Type	Parking_Space	Security	Electricity	Furnished	Security_Doors	CCTV	Pool	Gym	BQ
0	yaba	1	1	2	700000.0	Mini flat	0	0	0	0	0	0	0	0	
1	yaba	1	1	2	700000.0	Mini flat	0	0	0	0	0	0	0	0	
2	yaba	1	1	2	650000.0	Mini flat	0	0	0	0	0	0	0	0	
3	yaba	1	1	1	450000.0	Mini flat	0	0	0	0	0	0	0	0	
4	yaba	3	3	4	800000.0	Detached duplex	0	1	0	0	0	0	0	0	

Select Target

The target in this data set is Price Target is our Prediction. In this data set we want to predict house price. y is the target, as seen below, if y is increasing, x will be increasing. X is independent feature while Y is the dependent. Y is the target.

In [34]:
```
y = lagos_houses.price
y.head()
```

Out[34]:
```
0    700000.0
1    700000.0
2    650000.0
3    450000.0
4    800000.0
Name: price, dtype: float64
```

Feature Selection:

This is neccesary because we will not be predicting with all the colums to predict our target. Only the column that are important to how house prices is determined. Another reason is because the categorical columns are not needed for the prediction so we will only be using the columns with integers.

- Feature Selection: This is where you choose the feature to train your model. We will call the training data 'X'

In [35]:
```
# We want to view the column
lagos_houses.columns
```

Out[35]:
```
Index(['location', 'bed', 'bath', 'toilet', 'price', 'Property_Type',
       'Parking_Space', 'Security', 'Electricity', 'Furnished',
       'Security_Doors', 'CCTV', 'Pool', 'Gym', 'BQ'],
      dtype='object')
```

Creating List of Features

We want to crate a list of features, for now they are usually numeriacl coloumns although they cloud be categorical. Note that price is the target and will not be included in the category

In [36]:
```
feature_columns = ['bed', 'bath', 'toilet',
       'Parking_Space', 'Security', 'Electricity', 'Furnished',
       'Security_Doors', 'CCTV', 'Pool', 'Gym', 'BQ']
```

Now we want to create X by filtering the housing data with selected features

In [37]:
```
X = lagos_houses[feature_columns]
X.head()
```

Out[37]:

	bed	bath	toilet	Parking_Space	Security	Electricity	Furnished	Security_Doors	CCTV	Pool	Gym	BQ
0	1	1	2	0	0	0	0	0	0	0	0	0
1	1	1	2	0	0	0	0	0	0	0	0	0
2	1	1	2	0	0	0	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0	0	0	0	0
4	3	3	4	0	1	0	0	0	0	0	0	0

MY FISRT MACHINE LEARNING MODEL

Importing

- I need to import from Scikit learn

In [38]:
```
from sklearn.tree import DecisionTreeRegressor
```

After importing decision tree, we want to create an instance using the data set.

- Note that: Random_State is used to reproduce your train in the same trend so as to keep getting the same result when you execute. You could use 40, 42, 35. in other words it is used to get the same train accross different execution

In [39]:
```
lagos_model = DecisionTreeRegressor(random_state = 42)
```

Model Fitting

- This is to esure your model fits in the exact patern you train it. Beloww is how to fit the model to data That is i want to train my model.

In [40]:
```
lagos_model.fit(X, y)
```

Out[40]:
```
DecisionTreeRegressor(random_state=42)
```

Predicitng House Prices Using ML

In [41]:
```
preds = lagos_model.predict(X)
```

Here we recall y contains actual prices and X contains the predicted prices.

In [42]:
```
print('Actual Prices ', y.head().tolist())
print('Predicted Prices ', preds[:5])
```

```
Actual Prices  [700000.0, 700000.0, 650000.0, 450000.0, 800000.0]
Predicted Prices  [ 714471.74447174  714471.74447174  714471.74447174  460478.95208071
 1477500.        ]
```

Perfomance Statistical Metrics

- This is used to evaluate the model.
  - We can use performance metrix to evaluate the model so that predictions will not be too close to the actual price.
    - EG: Mean Absolute Error (MAE) As we will see below:

Let us assume the house is N750, 000 (Naira) and our model predicts N800, 000 (Naira). The difference is N50, 000(Naira) we can then call that 'error' or 'Residual Errors'

MAE

is the mean of all such errors in the range of the predicted values. It is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. MAE is calculated as the sum of absolute errors divided by the sample size.

Mean Square Error MSE

this is another statistical metrics used to evalute the errors this squares all the errors and finds the avg

Residual Square Error R2_Score

This also gives a Square of the prediction and then Scores it in percentile

In [43]:
```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y, preds)
print(mae)
```

```
239587.33686843375
```

In [44]:
```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y, preds)
print(mse)
```

```
121988805608.65668
```

In [45]:
```
from sklearn.metrics import r2_score
score = r2_score(y, preds)
print(score)
```

```
0.4460245589970214
```

Validation

- This invloves the following:
  - TRAIN
  - TEST
  - SPLIT
    - These mean when you split your data into a training test and testing set based on a particular ratio. The ratio can either be 30:70 or 40:60. The training set is used for training the model, and the testing set is used to test your models. These allows you to train your model on the training set, and test their accuracy on the unseen testing set.

  To achive this we will import train, test from SK Learning

In [46]:
```
from sklearn.model_selection import train_test_split
```

We want to use 30% to train our model

In [47]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

In [48]:
```
print('X_train: ', X_train.shape)
print('Y_train: ', y_train.shape)
print('X_train: ', X_test.shape)
print('Y_train: ', y_test.shape)
```

```
X_train:  (3735, 12)
Y_train:  (3735,)
X_train:  (1601, 12)
Y_train:  (1601,)
```

We want to retein the model, we might retain the initial training of Lagos_model or change it.

In [49]:
```
lagos_model.fit(X_train, y_train)
```

Out[49]:
```
DecisionTreeRegressor(random_state=42)
```

In [50]:
```
preds2 = lagos_model.predict(X_test)
```

In [71]:
```
# Mannually compare

print(y_test.head().tolist())
print(preds2[:5])
```

```
[450000.0, 900000.0, 400000.0, 500000.0, 400000.0]
[464017.85714286 623050.84745763 461739.87853277 461739.87853277
 461739.87853277]
```

In [51]:
```
# get the mean absolute error
mae2 = mean_absolute_error(y_test, preds2)
print('Without Spliting ', mae)
print('With Spliting ', mae2)
```

```
Without Spliting  239587.33686843375
With Spliting  242387.07752908807
```

In [52]:
```
from sklearn.metrics import r2_score
score2 = r2_score(y_test, preds2)
print(score2)
```

```
0.38115651788647853
```

Observation

As seen compare to the earlier prediction, using the split method gave a close prediction.

Random Forest Regressor

In [53]:
```
from sklearn.ensemble import RandomForestRegressor
lagos_RF_model = RandomForestRegressor()
```

In [54]:
```
# We now want to fit it
lagos_RF_model.fit(X_train, y_train)
```

Out[54]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [55]:
```
# Now we want.to predict on the test data
preds_RF = lagos_RF_model.predict(X_test)
```

Now we want to mannually compare

In [56]:
```
print(y_test.head().tolist())
print(preds_RF[:5])
```

```
[450000.0, 900000.0, 400000.0, 500000.0, 400000.0]
[466905.16018469 619923.9394551  461288.53932123 461288.53932123
 461288.53932123]
```

Calculating the Errors

In [57]:
```
mae_RF = mean_absolute_error(y_test, preds_RF)
print('Decision Tree With TRAIN TEST SPLIT ', mae2)
print('Random Forest with TRAIN TEST SPLIT  ', mae_RF)
```

```
Decision Tree With TRAIN TEST SPLIT  242387.07752908807
Random Forest with TRAIN TEST SPLIT   241524.55812917763
```

Squard for both Decision Tree and Random Forest

In [58]:
```
score_rf = r2_score(y_test, preds_RF)
print('RandomForest Tree With TRAIN TEST SPLIT ', score_rf)
print('DecisionTree with TRAIN TEST SPLIT  ', score2)
```

```
RandomForest Tree With TRAIN TEST SPLIT  0.38946780812279924
DecisionTree with TRAIN TEST SPLIT   0.38115651788647853
```

Summary

- It is observed that decision tree algorithm performs better compared to random forest with aproximately 65% accuracy after the performance metrics was done on them

END