Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("mobile")
```

To display top 10 rows

In [3]:
```
1 df.head(10)
```

Out[3]:

	Phone Name	Rating ?/5	Number of Ratings	RAM	ROM/Storage	Back/Rare Camera	Front Camera	Battery	Processor	Price in INR	Date of Scraping
0	POCO C50 (Royal Blue, 32 GB)	4.2	33,561	2 GB RAM	32 GB ROM	8MP Dual Camera	5MP Front Camera	5000 mAh	Mediatek Helio A22 Processor, Upto 2.0 GHz Pro...	₹5,649	2023-06-17
1	POCO M4 5G (Cool Blue, 64 GB)	4.2	77,128	4 GB RAM	64 GB ROM	50MP + 2MP	8MP Front Camera	5000 mAh	Mediatek Dimensity 700 Processor	₹11,999	2023-06-17
2	POCO C51 (Royal Blue, 64 GB)	4.3	15,175	4 GB RAM	64 GB ROM	8MP Dual Rear Camera	5MP Front Camera	5000 mAh	Helio G36 Processor	₹6,999	2023-06-17
3	POCO C55 (Cool Blue, 64 GB)	4.2	22,621	4 GB RAM	64 GB ROM	50MP Dual Rear Camera	5MP Front Camera	5000 mAh	Mediatek Helio G85 Processor	₹7,749	2023-06-17
4	POCO C51 (Power Black, 64 GB)	4.3	15,175	4 GB RAM	64 GB ROM	8MP Dual Rear Camera	5MP Front Camera	5000 mAh	Helio G36 Processor	₹6,999	2023-06-17
5	POCO M4 5G (Power Black, 64 GB)	4.2	77,128	4 GB RAM	64 GB ROM	50MP + 2MP	8MP Front Camera	5000 mAh	Mediatek Dimensity 700 Processor	₹11,999	2023-06-17
6	POCO C55 (Power Black, 64 GB)	4.2	22,621	4 GB RAM	64 GB ROM	50MP Dual Rear Camera	5MP Front Camera	5000 mAh	Mediatek Helio G85 Processor	₹7,749	2023-06-17
7	POCO C55 (Forest Green, 64 GB)	4.2	22,621	4 GB RAM	64 GB ROM	50MP Dual Rear Camera	5MP Front Camera	5000 mAh	Mediatek Helio G85 Processor	₹7,749	2023-06-17
8	POCO C55 (Cool Blue, 128 GB)	4.1	13,647	6 GB RAM	128 GB ROM	50MP Dual Rear Camera	5MP Front Camera	5000 mAh	Mediatek Helio G85 Processor	₹9,249	2023-06-17
9	POCO M4 5G (Yellow, 128 GB)	4.2	40,525	6 GB RAM	128 GB ROM	50MP + 2MP	8MP Front Camera	5000 mAh	Mediatek Dimensity 700 Processor	₹13,999	2023-06-17

Data Cleaning And Pre-Processing

In [4]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1836 entries, 0 to 1835
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Phone Name          1836 non-null   object 
 1   Rating ?/5          1836 non-null   float64
 2   Number of Ratings   1836 non-null   object 
 3   RAM                 1836 non-null   object 
 4   ROM/Storage         1662 non-null   object 
 5   Back/Rare Camera    1827 non-null   object 
 6   Front Camera        1435 non-null   object 
 7   Battery             1826 non-null   object 
 8   Processor           1781 non-null   object 
 9   Price in INR        1836 non-null   object 
 10  Date of Scraping    1836 non-null   object 
dtypes: float64(1), object(10)
memory usage: 157.9+ KB
```

In [5]:
```
1 # Display the statistical summary
2 df.describe()
```

Out[5]:

	Rating ?/5
count	1836.000000
mean	4.210512
std	0.543912
min	0.000000
25%	4.200000
50%	4.300000
75%	4.400000
max	4.800000

In [6]:
```
# To display the col headings
df.columns
```

Out[6]:
```
Index(['Phone Name', 'Rating ?/5', 'Number of Ratings', 'RAM', 'ROM/Storage',
       'Back/Rare Camera', 'Front Camera', 'Battery', 'Processor',
       'Price in INR', 'Date of Scraping'],
      dtype='object')
```

In [7]:
```
cols=df.dropna(axis=1)
```

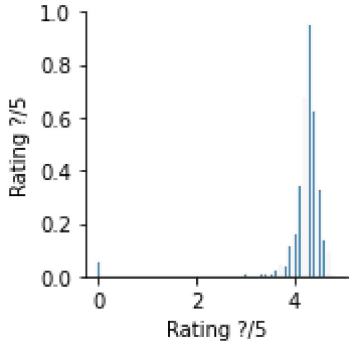
In [8]:
```
cols.columns
```

Out[8]:
```
Index(['Phone Name', 'Rating ?/5', 'Number of Ratings', 'RAM', 'Price in INR',
       'Date of Scraping'],
      dtype='object')
```

EDA and Visualization

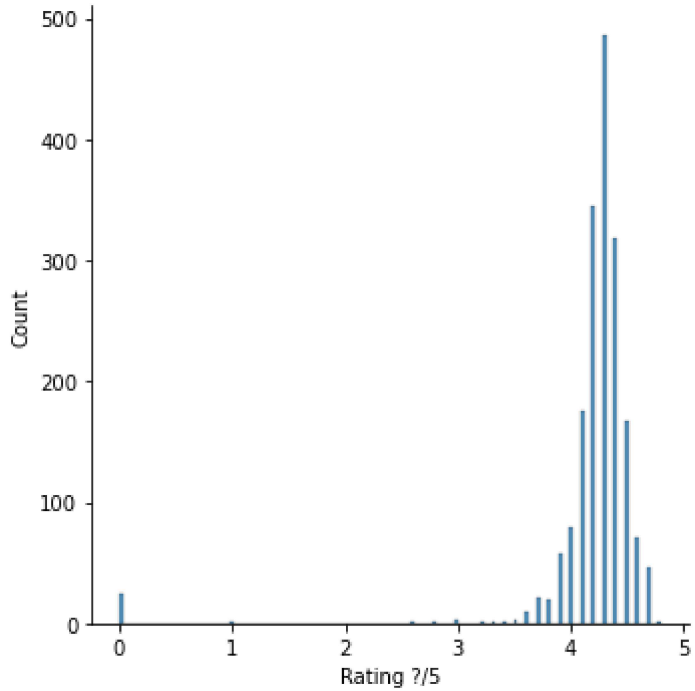
In [9]:
```
sns.pairplot(cols)
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x2117d7e5430>
```



In [10]:
```
1 sns.displot(df['Rating ?/5'])
```

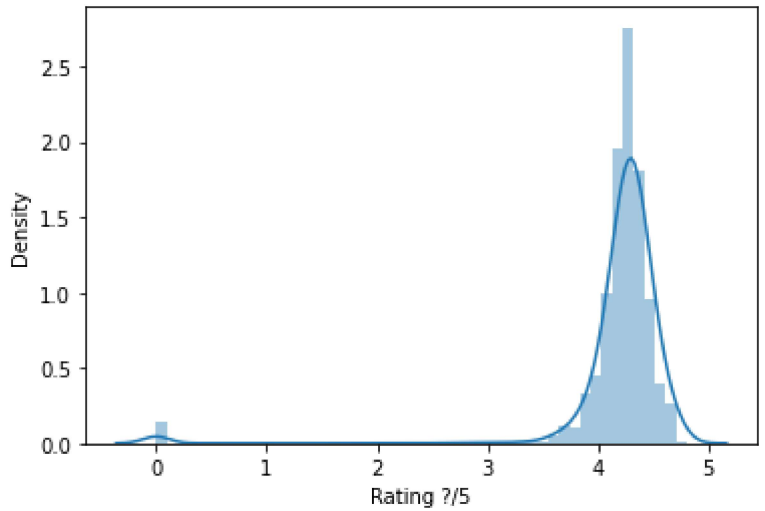
Out[10]: <seaborn.axisgrid.FacetGrid at 0x2117d939af0>



In [11]:
```
# We use displot in older version we get distplot use displot
sns.distplot(df['Rating ?/5'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[11]: <AxesSubplot:xlabel='Rating ?/5', ylabel='Density'>



In [12]:

```
df1=df[['Phone Name', 'Rating ?/5', 'Number of Ratings', 'RAM']]
df1
```

Out[12]:

	Phone Name	Rating ?/5	Number of Ratings	RAM
0	POCO C50 (Royal Blue, 32 GB)	4.2	33,561	2 GB RAM
1	POCO M4 5G (Cool Blue, 64 GB)	4.2	77,128	4 GB RAM
2	POCO C51 (Royal Blue, 64 GB)	4.3	15,175	4 GB RAM
3	POCO C55 (Cool Blue, 64 GB)	4.2	22,621	4 GB RAM
4	POCO C51 (Power Black, 64 GB)	4.3	15,175	4 GB RAM
...	...	...	...	...
1831	Infinix Note 7 (Forest Green, 64 GB)	4.3	25,582	4 GB RAM
1832	Infinix Note 7 (Bolivia Blue, 64 GB)	4.3	25,582	4 GB RAM
1833	Infinix Note 7 (Aether Black, 64 GB)	4.3	25,582	4 GB RAM
1834	Infinix Zero 8i (Silver Diamond, 128 GB)	4.2	7,117	8 GB RAM
1835	Infinix S5 (Quetzal Cyan, 64 GB)	4.3	15,701	4 GB RAM

1836 rows × 4 columns

In [13]:
```
1 sns.heatmap(df1.corr())
```

Out[13]: <AxesSubplot:>

To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [14]:
```
1 x=df1[['Rating ?/5']]
2 y=df1[['Rating ?/5']]
```

To split the dataset into test data

In [15]:
```
1 # importing lib for splitting test data
2 from sklearn.model_selection import train_test_split
```

In [16]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [17]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[17]:
```
LinearRegression()
```

In [18]:
```
print(lr.intercept_)
```

```
[-8.8817842e-16]
```

In [19]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

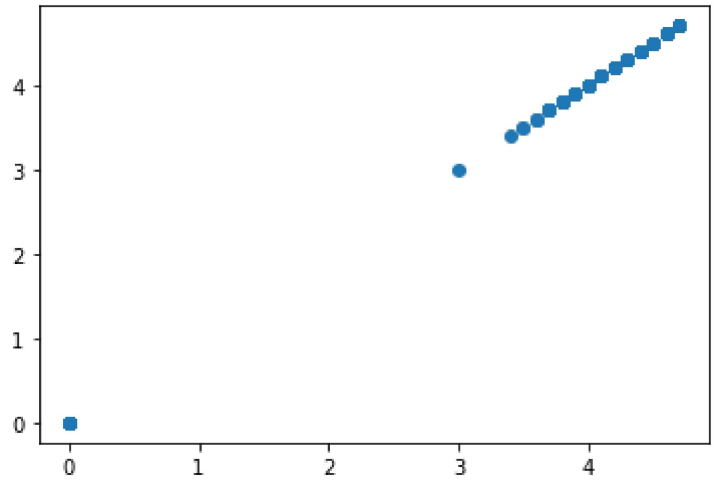
In [20]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[20]:

	0
0	1.0

In [21]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```

Out[21]: <matplotlib.collections.PathCollection at 0x2117fde2220>



In [22]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [23]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[23]: Ridge(alpha=10)

In [24]:
```
rr.score(x_test,y_test)
```

Out[24]: 0.9992583251780627

In [25]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[25]: Lasso(alpha=10)

In [26]:
```
la.score(x_test,y_test)
```

Out[26]: -0.0011523996088913524

ELASTIC NET

In [27]:
```python
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

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

In [28]:
```python
print(en.coef_)
```

```
[0.]
```

In [29]:
```python
print(en.intercept_)
```

```
[4.21642023]
```

In [30]:
```python
prediction=en.predict(x_test)
prediction
```

Out[30]:
```
array([4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
       4.21642023, 4.21642023, 4.21642023, 4.21642023, 4.21642023,
```

In [31]:
```
print(en.score(x_test,y_test))
```

```
-0.0011523996088913524
```

EVALUATION METRICS

In [32]:
```
from sklearn import metrics
```

In [33]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.22953342701985072
```

In [34]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.33671084519305483
```

In [35]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 0.5802679081192194
```