In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore
from scipy import stats
from scipy.stats import skew
from sklearn.model_selection import train_test_split, GridSearchCV ,RandomizedSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder, StandardScaler
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_
auc_score, confusion_matrix
```

Read Dataset

In [2]:

```
df = pd.read_csv('used_device_data.csv')
```

In [3]:

```
df
```

Out[3]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weigh
0	Honor	Android	14.50	yes	no	13.0	5.0	64.0	3.0	3020.0	146.
1	Honor	Android	17.30	yes	yes	13.0	16.0	128.0	8.0	4300.0	213.
2	Honor	Android	16.69	yes	yes	13.0	8.0	128.0	8.0	4200.0	213.
3	Honor	Android	25.50	yes	yes	13.0	8.0	64.0	6.0	7250.0	480.
4	Honor	Android	15.32	yes	no	13.0	8.0	64.0	3.0	5000.0	185.
...	...	...	...	...	...	...	...	...	...	...	.
3449	Asus	Android	15.34	yes	no	NaN	8.0	64.0	6.0	5000.0	190.
3450	Asus	Android	15.24	yes	no	13.0	8.0	128.0	8.0	4000.0	200.
3451	Alcatel	Android	15.80	yes	no	13.0	5.0	32.0	3.0	4000.0	165.
3452	Alcatel	Android	15.80	yes	no	13.0	5.0	32.0	2.0	4000.0	160.
3453	Alcatel	Android	12.83	yes	no	13.0	5.0	16.0	2.0	4000.0	168.

3454 rows × 15 columns

In [ ]:

```
'''device_brand: Name of manufacturing brand
os: OS on which the device runs
screen_size: Size of the screen in cm
4g: Whether 4G is available or not
5g: Whether 5G is available or not
front_camera_mp: Resolution of the rear camera in megapixels
back_camera_mp: Resolution of the front camera in megapixels
internal_memory: Amount of internal memory (ROM) in GB
```

```
ram: Amount of RAM in GB
battery: Energy capacity of the device battery in mAh
weight: Weight of the device in grams
release_year: Year when the device model was released
days_used: Number of days the used/refurbished device has been used
normalized_new_price: Normalized price of a new device of the same model
normalized_used_price (TARGET): Normalized price of the used/refurbished device'''
```

Null Values

In [4]:

```
df.isnull().sum()
```

Out[4]:

```
device_brand              0
os                        0
screen_size               0
4g                        0
5g                        0
rear_camera_mp          179
front_camera_mp           2
internal_memory           4
ram                       4
battery                   6
weight                    7
release_year              0
days_used                 0
normalized_used_price     0
normalized_new_price      0
dtype: int64
```

Mean, Median, Mode, Variance, Standard Deviation

In [5]:

```
mean_value = df['rear_camera_mp'].mean()
mean_value
print('Mean value of rear_camera_mp is : ', mean_value)

median_value = df['rear_camera_mp'].median()  # Middle Value of rear camera megapixles co
lumn
median_value
print ('Median Value of rear camera mp is  : ',median_value)

mode_value = df['rear_camera_mp'].mode()  # Most occuring value in rear camera megapixles
column
print(f"Mode rear camera mp : {mode_value.values.tolist()}")
mode_counts = df['rear_camera_mp'].value_counts()

variance_value = df['rear_camera_mp'].var()  # Measure How datapoints differ from the mean
variance_value
print ('Variance value of rear camera is : ',variance_value)

Std_dev_value = df['rear_camera_mp'].std()  # Measure How scattered the data is in relatio
n to the mean
Std_dev_value
print ('Standard deviation value of rear camera mp  is : ',Std_dev_value)
```

```
Mean value of rear_camera_mp is :   9.460207633587787
Median Value of rear camera mp is  :  8.0
Mode rear camera mp : [13.0]
Variance value of rear camera is :  23.188666969703434
Standard deviation value of rear camera mp  is :  4.81546124163651
```

Filling Null values with mean value

In [6]:

```
df['rear_camera_mp'].fillna(mean_value, inplace=True)
# The data is modified in place, which means it will return nothing and the dataframe is now updated.
```

In [7]:

```
df
```

Out[7]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weigh
0	Honor	Android	14.50	yes	no	13.000000	5.0	64.0	3.0	3020.0	146.
1	Honor	Android	17.30	yes	yes	13.000000	16.0	128.0	8.0	4300.0	213.
2	Honor	Android	16.69	yes	yes	13.000000	8.0	128.0	8.0	4200.0	213.
3	Honor	Android	25.50	yes	yes	13.000000	8.0	64.0	6.0	7250.0	480.
4	Honor	Android	15.32	yes	no	13.000000	8.0	64.0	3.0	5000.0	185.
...	...	...	...	...	...	...	...	...	...	...	.
3449	Asus	Android	15.34	yes	no	9.460208	8.0	64.0	6.0	5000.0	190.
3450	Asus	Android	15.24	yes	no	13.000000	8.0	128.0	8.0	4000.0	200.
3451	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	3.0	4000.0	165.
3452	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	2.0	4000.0	160.
3453	Alcatel	Android	12.83	yes	no	13.000000	5.0	16.0	2.0	4000.0	168.

3454 rows × 15 columns

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
device_brand             0
os                       0
screen_size              0
4g                       0
5g                       0
rear_camera_mp           0
front_camera_mp          2
internal_memory          4
ram                      4
battery                  6
weight                   7
release_year             0
days_used                0
normalized_used_price    0
normalized_new_price     0
dtype: int64
```

Removing All Null Values from Dataset

In [9]:

```
drop1 = df.dropna(subset=['front_camera_mp'], inplace=True)
drop1

drop = df.dropna(subset=['internal_memory'], inplace=True)
drop
```

```
drop3 = df.dropna(subset=['ram'], inplace=True)
drop3

drop4 = df.dropna(subset=['battery'], inplace=True)
drop4

drop5 = df.dropna(subset=['weight'], inplace=True)
drop5
```

In [10]:

```
df.isnull().sum()
```

Out[10]:

```
device_brand             0
os                       0
screen_size              0
4g                       0
5g                       0
rear_camera_mp           0
front_camera_mp          0
internal_memory          0
ram                      0
battery                  0
weight                   0
release_year             0
days_used                0
normalized_used_price    0
normalized_new_price     0
dtype: int64
```

Information about Dataset

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3432 entries, 0 to 3453
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   device_brand           3432 non-null   object 
 1   os                     3432 non-null   object 
 2   screen_size            3432 non-null   float64
 3   4g                     3432 non-null   object 
 4   5g                     3432 non-null   object 
 5   rear_camera_mp         3432 non-null   float64
 6   front_camera_mp        3432 non-null   float64
 7   internal_memory        3432 non-null   float64
 8   ram                    3432 non-null   float64
 9   battery                3432 non-null   float64
 10  weight                 3432 non-null   float64
 11  release_year           3432 non-null   int64  
 12  days_used              3432 non-null   int64  
 13  normalized_used_price  3432 non-null   float64
 14  normalized_new_price   3432 non-null   float64
dtypes: float64(9), int64(2), object(4)
memory usage: 429.0+ KB
```

How many Rows and Columns are present in Dataset?

In [82]:

```
df.shape
```

Out[82]:

```
(3432, 15)
```

Column Names

In [83]:

```
df.columns
```

Out[83]:

```
Index(['device_brand', 'os', 'screen_size', '4g', '5g', 'rear_camera_mp',
       'front_camera_mp', 'internal_memory', 'ram', 'battery', 'weight',
       'release_year', 'days_used', 'normalized_used_price',
       'normalized_new_price'],
      dtype='object')
```

Description of the Dataset

In [84]:

```
df.describe()
```

Out[84]:

	screen_size	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight	release_year
count	3432.000000	3432.000000	3432.000000	3432.000000	3432.000000	3432.000000	3432.000000	3432.000000
mean	13.733686	9.475512	6.582197	54.742672	4.042107	3139.037733	182.870455	2015.966492
std	3.788795	4.675254	6.979159	85.151126	1.360061	1298.889825	88.081369	2.299186
min	5.080000	0.080000	0.000000	0.010000	0.020000	500.000000	69.000000	2013.000000
25%	12.700000	5.000000	2.000000	16.000000	4.000000	2100.000000	142.000000	2014.000000
50%	12.830000	9.460208	5.000000	32.000000	4.000000	3000.000000	160.000000	2016.000000
75%	15.370000	13.000000	8.000000	64.000000	4.000000	4000.000000	185.000000	2018.000000
max	30.710000	48.000000	32.000000	1024.000000	12.000000	9720.000000	855.000000	2020.000000

Unique Values

In [85]:

```
df['device_brand'].unique()
```

Out[85]:

```
array(['Honor', 'Others', 'HTC', 'Huawei', 'Infinix', 'Lava', 'Lenovo',
       'LG', 'Meizu', 'Micromax', 'Motorola', 'Nokia', 'OnePlus', 'Oppo',
       'Realme', 'Samsung', 'Vivo', 'Xiaomi', 'ZTE', 'Apple', 'Asus',
       'Coolpad', 'Acer', 'Alcatel', 'BlackBerry', 'Celkon', 'Gionee',
       'Google', 'Karbonn', 'Microsoft', 'Panasonic', 'Sony', 'Spice',
       'XOLO'], dtype=object)
```

In [86]:

```
df['os'].unique()
```

Out[86]:

```
array(['Android', 'Others', 'iOS', 'Windows'], dtype=object)
```

In [87]:

```
df['release_year'].unique()
```

Out[87]:

```
array([2020, 2019, 2013, 2014, 2016, 2018, 2015, 2017], dtype=int64)
```

Head

In [88]:

```
df.head(10)
```

Out[88]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
0	Honor	Android	14.50	yes	no	13.0	5.0	64.0	3.0	3020.0	146.0
1	Honor	Android	17.30	yes	yes	13.0	16.0	128.0	8.0	4300.0	213.0
2	Honor	Android	16.69	yes	yes	13.0	8.0	128.0	8.0	4200.0	213.0
3	Honor	Android	25.50	yes	yes	13.0	8.0	64.0	6.0	7250.0	480.0
4	Honor	Android	15.32	yes	no	13.0	8.0	64.0	3.0	5000.0	185.0
5	Honor	Android	16.23	yes	no	13.0	8.0	64.0	4.0	4000.0	176.0
6	Honor	Android	13.84	yes	no	8.0	5.0	32.0	2.0	3020.0	144.0
7	Honor	Android	15.77	yes	no	13.0	8.0	64.0	4.0	3400.0	164.0
8	Honor	Android	15.32	yes	no	13.0	16.0	128.0	6.0	4000.0	165.0
9	Honor	Android	16.23	yes	no	13.0	8.0	128.0	6.0	4000.0	176.0

Tail

In [89]:

```
df.tail(10)
```

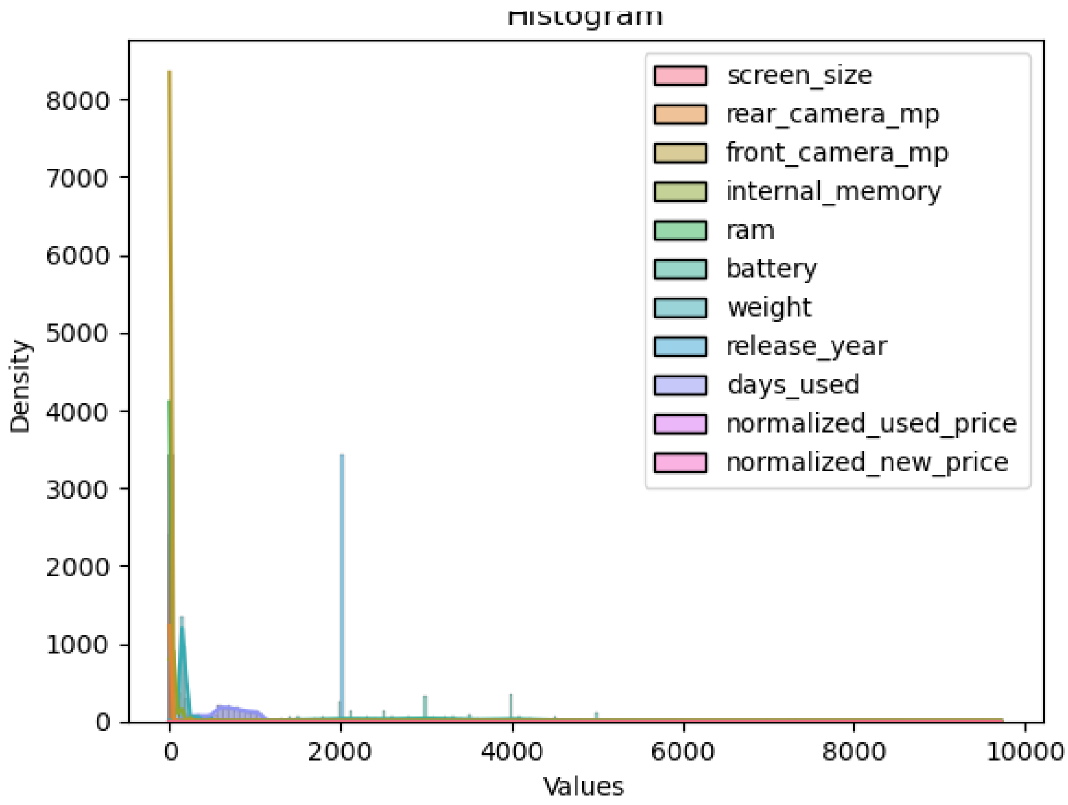
Out[89]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
3444	Apple	iOS	10.34	yes	no	12.000000	7.0	64.0	3.0	1821.0	148.0
3445	Apple	iOS	15.37	yes	no	8.000000	7.0	64.0	4.0	3969.0	226.0
3446	Apple	iOS	12.90	yes	no	8.000000	7.0	64.0	4.0	3046.0	188.0
3447	Apple	iOS	15.27	yes	no	8.000000	7.0	64.0	4.0	3110.0	194.0
3448	Asus	Android	16.74	yes	no	9.460208	24.0	128.0	8.0	6000.0	240.0
3449	Asus	Android	15.34	yes	no	9.460208	8.0	64.0	6.0	5000.0	190.0
3450	Asus	Android	15.24	yes	no	13.000000	8.0	128.0	8.0	4000.0	200.0
3451	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	3.0	4000.0	165.0
3452	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	2.0	4000.0	160.0
3453	Alcatel	Android	12.83	yes	no	13.000000	5.0	16.0	2.0	4000.0	168.0

In [90]:

```python
a = sns.histplot(df,kde = True, color = 'skyblue')
plt.title('Histogram')
plt.xlabel('Values')
plt.ylabel('Density')
plt.show()
```

Histogram

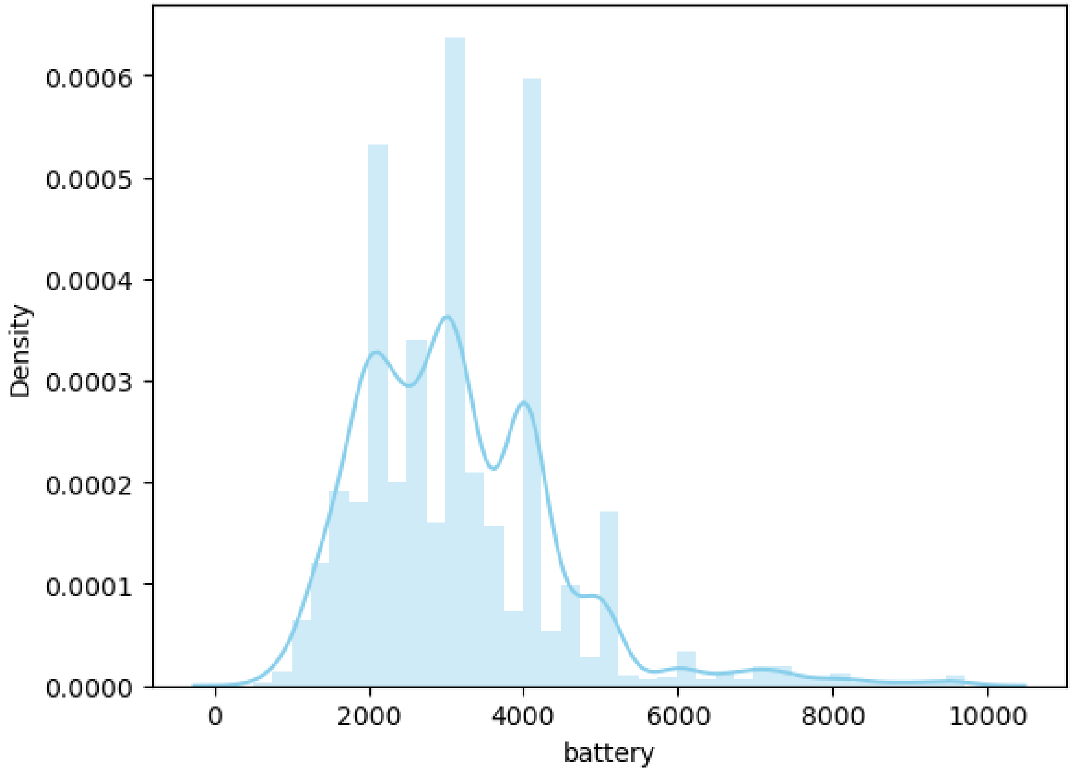


In [91]:

```
sns.distplot(df['battery'], kde = True, color = 'skyblue')
```

Out[91]:

```
<Axes: xlabel='battery', ylabel='Density'>
```



Duplicate Values

In [15]:

```
duplicates = df[df.duplicated('ram', keep=False)] # keep = False means Mark all duplicat
e values are True.
```

In [16]:

```
duplicates
```

Out[16]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weigh
0	Honor	Android	14.50	yes	no	13.000000	5.0	64.0	3.0	3020.0	146.
1	Honor	Android	17.30	yes	yes	13.000000	16.0	128.0	8.0	4300.0	213.
2	Honor	Android	16.69	yes	yes	13.000000	8.0	128.0	8.0	4200.0	213.
3	Honor	Android	25.50	yes	yes	13.000000	8.0	64.0	6.0	7250.0	480.
4	Honor	Android	15.32	yes	no	13.000000	8.0	64.0	3.0	5000.0	185.
...	...	...	...	...	...	...	...	...	...	...	.
3449	Asus	Android	15.34	yes	no	9.460208	8.0	64.0	6.0	5000.0	190.
3450	Asus	Android	15.24	yes	no	13.000000	8.0	128.0	8.0	4000.0	200.
3451	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	3.0	4000.0	165.
3452	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	2.0	4000.0	160.
3453	Alcatel	Android	12.83	yes	no	13.000000	5.0	16.0	2.0	4000.0	168.

3431 rows × 15 columns

In [17]:

```
count_dup = duplicates['ram'].value_counts()
```

In [18]:

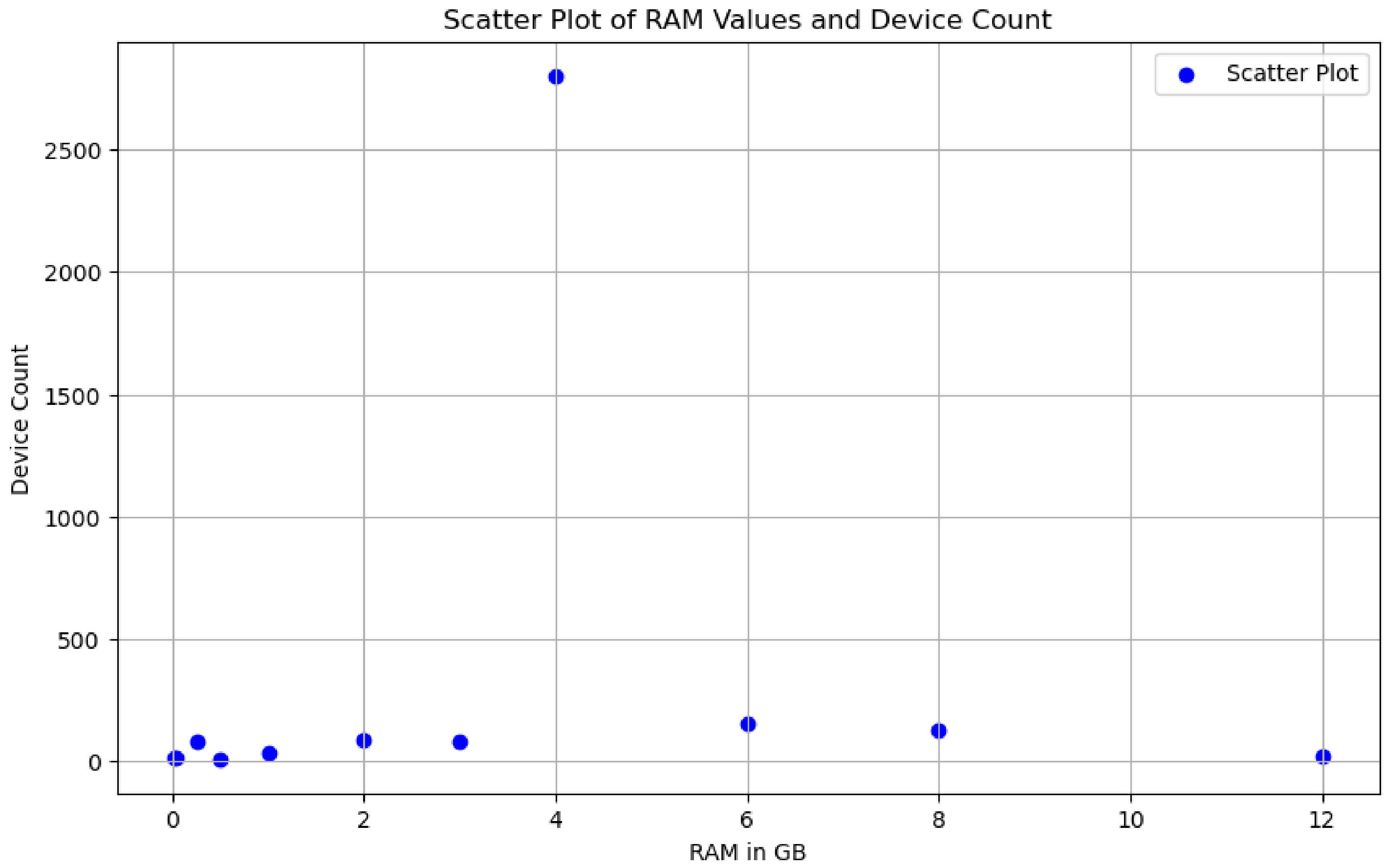
```
count_dup
```

Out[18]:

```
ram
4.00     2802
6.00      154
8.00      130
2.00       90
0.25       83
3.00       81
1.00       34
12.00      18
0.03       16
0.02       14
0.50        9
Name: count, dtype: int64
```

In [19]:

```
count_dup = duplicates['ram'].value_counts()

# Get labels and counts
labels = count_dup.index
counts = count_dup.values

# Create a scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(labels, counts, color='blue', label='Scatter Plot')
plt.xlabel('RAM in GB')
plt.ylabel('Device Count')
plt.title('Scatter Plot of RAM Values and Device Count')
plt.legend()
plt.grid()

# Show the scatter plot
plt.show()
```



In [ ]:

```
Observation :
    2802 devices have 4 GB RAM
    9 devices have 0.50 GB RAM
    Only 18 devices have 12 GB which highest capacity RAM
```

In [98]:

```python
highest_capacity_ram = df[df['ram'] == 12.00]
highest_capacity_ram
```

Out[98]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weigh
44	Huawei	Android	16.59	yes	yes	13.000000	16.0	512.0	12.0	4500.0	198.
109	Motorola	Android	15.42	yes	yes	9.460208	25.0	256.0	12.0	5000.0	203.
120	OnePlus	Android	16.94	yes	yes	9.460208	16.0	256.0	12.0	4085.0	206.
198	Xiaomi	Android	20.12	yes	yes	12.000000	20.0	512.0	12.0	4050.0	241.
263	Huawei	Android	16.59	yes	yes	13.000000	16.0	512.0	12.0	4500.0	198.
328	Motorola	Android	15.42	yes	yes	9.460208	25.0	256.0	12.0	5000.0	203.
339	OnePlus	Android	16.94	yes	yes	9.460208	16.0	256.0	12.0	4085.0	206.
372	Samsung	Android	15.32	yes	yes	12.000000	10.0	256.0	12.0	3500.0	168.
3250	Oppo	Android	15.37	yes	yes	9.460208	32.0	256.0	12.0	4025.0	171.
3252	Oppo	Android	15.42	yes	yes	9.460208	32.0	256.0	12.0	4260.0	217.
3391	Oppo	Android	15.37	yes	yes	9.460208	32.0	256.0	12.0	4025.0	171.
3393	Oppo	Android	15.42	yes	yes	9.460208	32.0	256.0	12.0	4260.0	217.
3420	Samsung	Android	15.47	yes	yes	8.000000	13.0	128.0	12.0	5000.0	222.
3421	Samsung	Android	15.47	yes	no	8.000000	13.0	128.0	12.0	5000.0	220.
3422	Samsung	Android	15.42	yes	yes	8.000000	13.0	128.0	12.0	4500.0	188.
3424	Samsung	Android	15.29	yes	yes	8.000000	13.0	128.0	12.0	4000.0	163.

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weigh
3436	Samsung	Android	17.86	yes	yes	12.000000	8.0	512.0	12.0	4235.0	263.
3440	Samsung	Android	15.44	yes	no	12.000000	10.0	256.0	12.0	4300.0	196.

In [99]:

```
lowest_capacity_ram= df[df['ram'] == 0.02]
lowest_capacity_ram
```

Out[99]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
116	Nokia	Others	5.18	no	no	0.30	0.0	0.06	0.02	1200.0	88.2
2044	Nokia	Others	5.18	yes	no	2.00	0.0	0.10	0.02	1100.0	117.0
2049	Nokia	Others	5.18	yes	no	2.00	0.0	0.06	0.02	1200.0	88.1
2052	Nokia	Others	5.18	no	no	2.00	0.0	0.10	0.02	1200.0	88.2
2057	Nokia	Others	5.18	no	no	2.00	0.0	0.10	0.02	1000.0	160.0
2060	Nokia	Others	5.28	no	no	2.00	0.0	0.06	0.02	1200.0	91.8
2062	Nokia	Others	5.18	no	no	2.00	0.0	0.10	0.02	1100.0	79.0
2065	Nokia	Others	5.18	no	no	0.30	0.0	0.06	0.02	1100.0	78.6
2074	Nokia	Others	5.28	no	no	2.00	0.0	0.10	0.02	1200.0	99.8
2084	Nokia	Others	5.18	no	no	2.00	0.0	0.10	0.02	1830.0	83.6
2098	Nokia	Others	5.18	no	no	1.30	0.0	0.06	0.02	1020.0	89.6
2102	Nokia	Others	7.62	no	no	3.15	0.0	0.06	0.02	1200.0	98.2
2106	Nokia	Others	5.18	no	no	3.15	0.0	0.10	0.02	1110.0	102.0
2107	Nokia	Others	7.62	no	no	2.00	0.0	0.10	0.02	1110.0	103.7

In [100]:

```
Android_data = df[df['os'] == 'Android']
```

In [101]:

```
Android_data
```

Out[101]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weigh
0	Honor	Android	14.50	yes	no	13.000000	5.0	64.0	3.0	3020.0	146.
1	Honor	Android	17.30	yes	yes	13.000000	16.0	128.0	8.0	4300.0	213.
2	Honor	Android	16.69	yes	yes	13.000000	8.0	128.0	8.0	4200.0	213.
3	Honor	Android	25.50	yes	yes	13.000000	8.0	64.0	6.0	7250.0	480.
4	Honor	Android	15.32	yes	no	13.000000	8.0	64.0	3.0	5000.0	185.
...	...	...	...	...	...	...	...	...	...	...	.
3449	Asus	Android	15.34	yes	no	9.460208	8.0	64.0	6.0	5000.0	190.
3450	Asus	Android	15.24	yes	no	13.000000	8.0	128.0	8.0	4000.0	200.
3451	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	3.0	4000.0	165.
3452	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	2.0	4000.0	160.
3453	Alcatel	Android	12.83	yes	no	13.000000	5.0	16.0	2.0	4000.0	168.

3203 rows × 15 columns

In [102]:

```
Android_data.os.count()
```

Out[102]:

```
3203
```

In [103]:

```
IOS_data = df[df['os'] == 'iOS']
```

In [104]:

```
IOS_data
```

Out[104]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight	rel
391	Apple	iOS	27.94	yes	no	12.0	7.0	64.0	4.0	7812.0	468.0	
392	Apple	iOS	18.01	yes	no	8.0	1.2	16.0	2.0	5124.0	299.0	
642	Apple	iOS	25.53	yes	no	8.0	7.0	64.0	4.0	3969.0	456.0	
643	Apple	iOS	18.01	yes	no	8.0	7.0	64.0	4.0	5124.0	300.5	
644	Apple	iOS	30.71	yes	no	12.0	7.0	1024.0	4.0	9720.0	631.0	
645	Apple	iOS	27.94	yes	no	12.0	7.0	1024.0	4.0	7812.0	468.0	
646	Apple	iOS	15.37	yes	no	12.0	7.0	64.0	4.0	3174.0	208.0	
647	Apple	iOS	12.90	yes	no	12.0	7.0	64.0	4.0	2658.0	177.0	
648	Apple	iOS	15.27	yes	no	12.0	7.0	64.0	4.0	2942.0	194.0	
649	Apple	iOS	23.04	yes	no	8.0	1.2	32.0	4.0	5493.0	469.0	
650	Apple	iOS	12.90	yes	no	12.0	7.0	64.0	4.0	2716.0	174.0	
651	Apple	iOS	12.83	yes	no	12.0	7.0	64.0	4.0	2691.0	202.0	
652	Apple	iOS	10.34	yes	no	12.0	7.0	64.0	4.0	1821.0	148.0	
653	Apple	iOS	30.71	yes	no	12.0	7.0	64.0	4.0	2256.0	677.0	
654	Apple	iOS	25.53	yes	no	12.0	7.0	64.0	4.0	8134.0	469.0	
655	Apple	iOS	23.04	yes	no	8.0	1.2	32.0	4.0	8827.0	469.0	
656	Apple	iOS	12.83	yes	no	12.0	7.0	32.0	4.0	2900.0	188.0	
657	Apple	iOS	10.34	yes	no	12.0	7.0	32.0	4.0	1960.0	138.0	
658	Apple	iOS	23.04	yes	no	12.0	5.0	32.0	4.0	7306.0	437.0	
659	Apple	iOS	10.16	yes	no	12.0	1.2	16.0	4.0	1624.0	113.0	
660	Apple	iOS	12.83	yes	no	12.0	5.0	16.0	4.0	2750.0	192.0	
661	Apple	iOS	10.34	yes	no	12.0	5.0	16.0	4.0	1715.0	143.0	
662	Apple	iOS	30.71	yes	no	8.0	1.2	32.0	4.0	3937.0	713.0	
663	Apple	iOS	18.01	yes	no	8.0	1.2	16.0	4.0	5124.0	299.0	
664	Apple	iOS	23.04	yes	no	8.0	1.2	16.0	4.0	7340.0	437.0	
665	Apple	iOS	18.01	yes	no	5.0	1.2	16.0	4.0	6470.0	331.0	
666	Apple	iOS	12.83	yes	no	8.0	1.2	16.0	4.0	2915.0	172.0	
667	Apple	iOS	10.34	yes	no	8.0	1.2	16.0	4.0	1810.0	129.0	
668	Apple	iOS	23.04	yes	no	5.0	1.2	16.0	4.0	8600.0	469.0	
669	Apple	iOS	18.01	yes	no	5.0	1.2	16.0	4.0	6470.0	331.0	
670	Apple	iOS	10.16	yes	no	8.0	1.2	16.0	4.0	1560.0	112.0	
671	Apple	iOS	10.16	yes	no	8.0	1.2	32.0	4.0	1510.0	132.0	
3444	Apple	iOS	10.34	yes	no	12.0	7.0	64.0	3.0	1821.0	148.0	

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight	rel
3445	Apple	iOS	15.37	yes	no	8.0	7.0	64.0	4.0	3969.0	226.0	
3446	Apple	iOS	12.90	yes	no	8.0	7.0	64.0	4.0	3046.0	188.0	
3447	Apple	iOS	15.27	yes	no	8.0	7.0	64.0	4.0	3110.0	194.0	

In [105]:

```
IOS_data.os.count()
```

Out[105]:

36

In [106]:

```
Windows_data = df[df['os'] == 'Windows']
```

In [107]:

```
Windows_data
```

Out[107]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weigl
428	Acer	Windows	12.83	yes	no	21.0	8.0	32.0	4.0	2870.0	150
438	Acer	Windows	10.16	no	no	5.0	2.0	16.0	4.0	1300.0	119
603	Others	Windows	10.16	no	no	5.0	0.3	16.0	4.0	1420.0	110
604	Others	Windows	12.70	no	no	8.0	2.0	32.0	4.0	2000.0	156
605	Others	Windows	10.34	no	no	8.0	2.0	32.0	4.0	1750.0	98.
...	...	...	...	...	...	...	...	...	...	...	.
2300	Others	Windows	10.16	no	no	5.0	0.3	16.0	4.0	2500.0	154
2556	Samsung	Windows	12.70	yes	no	13.0	2.0	16.0	4.0	2600.0	135
2613	Samsung	Windows	12.12	yes	no	8.0	1.2	16.0	4.0	2000.0	144
2648	Samsung	Windows	10.16	yes	no	5.0	1.2	32.0	4.0	2100.0	125
3018	XOLO	Windows	10.34	no	no	8.0	2.0	32.0	4.0	1800.0	100

65 rows × 15 columns

In [108]:

```
Windows_data.os.count()
```

Out[108]:

65

In [109]:

```
Others_data = df[df['os'] == 'Others']
Others_data
```

Out[109]:

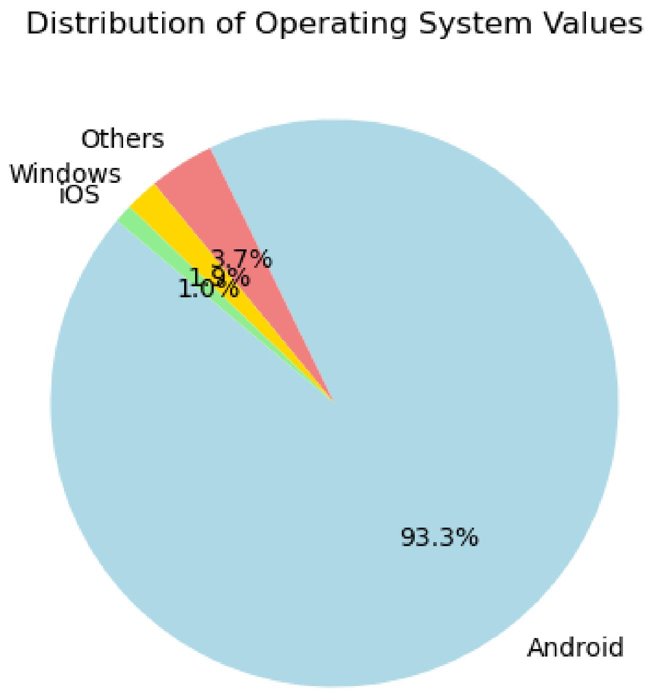
	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
78	LG	Others	5.28	yes	no	2.00	13.0	8.00	1.00	1470.0	127.0
113	Nokia	Others	5.18	no	no	0.30	0.0	0.10	0.03	1020.0	90.5
116	Nokia	Others	5.18	no	no	0.30	0.0	0.06	0.02	1200.0	88.2
297	LG	Others	5.28	yes	no	2.00	13.0	8.00	1.00	1470.0	127.0

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
332	Nokia	Others	5.18	no	no	0.30	0.0	0.06	0.03	1020.0	90.5
...	...	...	...	...	...	...	...	...	...	...	...
2796	Others	Others	5.18	no	no	1.30	2.0	128.00	0.25	2100.0	98.0
2802	Others	Others	5.13	no	no	2.00	2.0	128.00	0.25	2100.0	110.0
3170	ZTE	Others	10.16	no	no	3.15	5.0	16.00	4.00	1400.0	125.0
3246	Nokia	Others	5.28	yes	no	2.00	0.0	0.06	0.03	1500.0	118.0
3387	Nokia	Others	5.28	yes	no	2.00	0.0	0.10	0.03	1500.0	118.0

128 rows × 15 columns

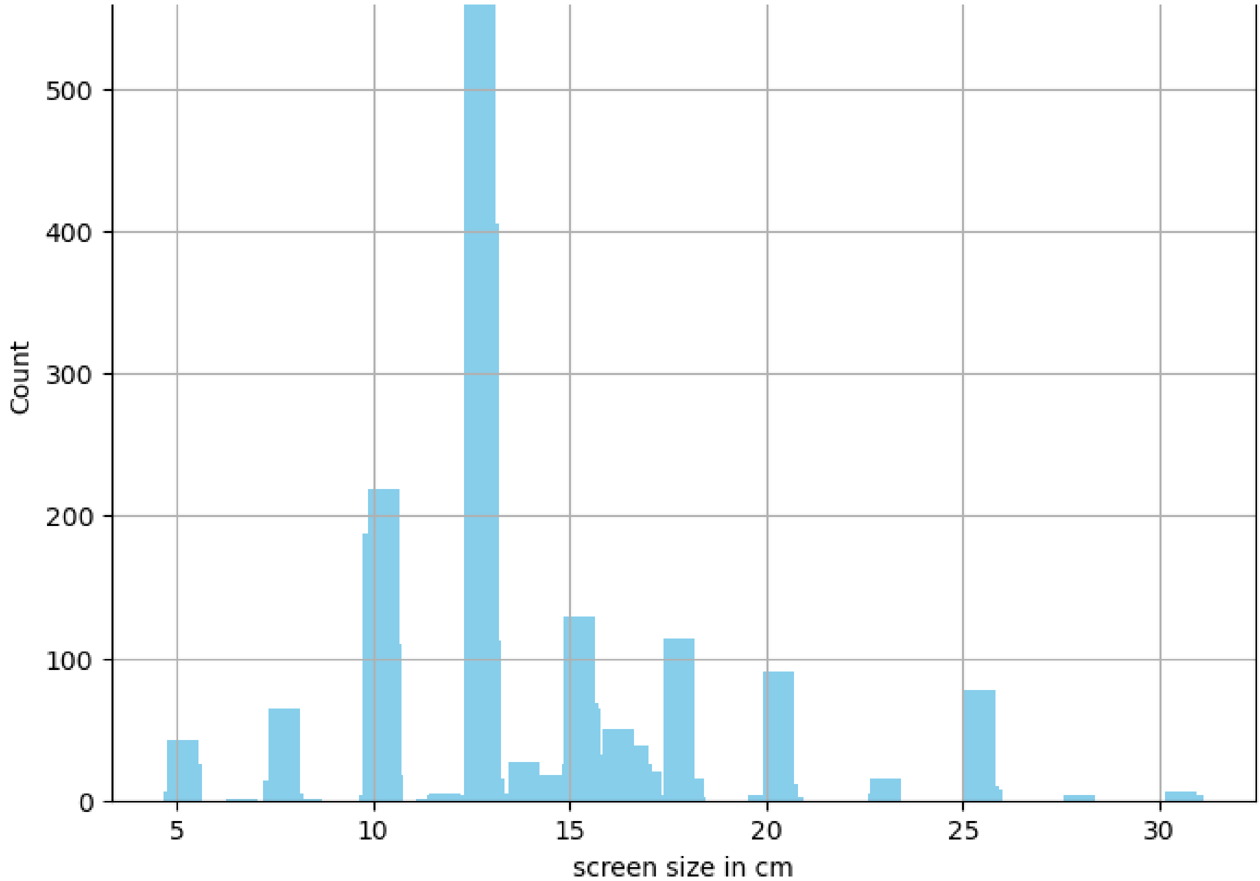
In [110]:

```
count_phones = df['os'].value_counts()


labels = count_phones.index
sizes = count_phones.values
colors = ['lightblue', 'lightcoral', 'gold', 'lightgreen']

plt.figure(figsize=(5, 5))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Operating System Values')
plt.show()
```



In [111]:

```
column_name = 'screen size in cm'
data = df['screen_size'].value_counts()

plt.figure(figsize=(8, 6))
plt.bar(data.index, data.values, color='skyblue')
plt.title(f'Bar Plot of {column_name}')
plt.xlabel(column_name)
plt.ylabel('Count')
plt.grid()
plt.show()
```



In [112]:

```
data_4g = df[df['4g'] == 'yes']
data_4g
```

Out[112]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weigh
0	Honor	Android	14.50	yes	no	13.000000	5.0	64.0	3.0	3020.0	146.
1	Honor	Android	17.30	yes	yes	13.000000	16.0	128.0	8.0	4300.0	213.
2	Honor	Android	16.69	yes	yes	13.000000	8.0	128.0	8.0	4200.0	213.
3	Honor	Android	25.50	yes	yes	13.000000	8.0	64.0	6.0	7250.0	480.
4	Honor	Android	15.32	yes	no	13.000000	8.0	64.0	3.0	5000.0	185.
...	...	...	...	...	...	...	...	...	...	...	.
3449	Asus	Android	15.34	yes	no	9.460208	8.0	64.0	6.0	5000.0	190.
3450	Asus	Android	15.24	yes	no	13.000000	8.0	128.0	8.0	4000.0	200.
3451	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	3.0	4000.0	165.
3452	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	2.0	4000.0	160.
3453	Alcatel	Android	12.83	yes	no	13.000000	5.0	16.0	2.0	4000.0	168.

2327 rows × 15 columns

In [113]:

```
data_4g1 = df[df['4g'] == 'no']
data_4g1
```

Out[113]:

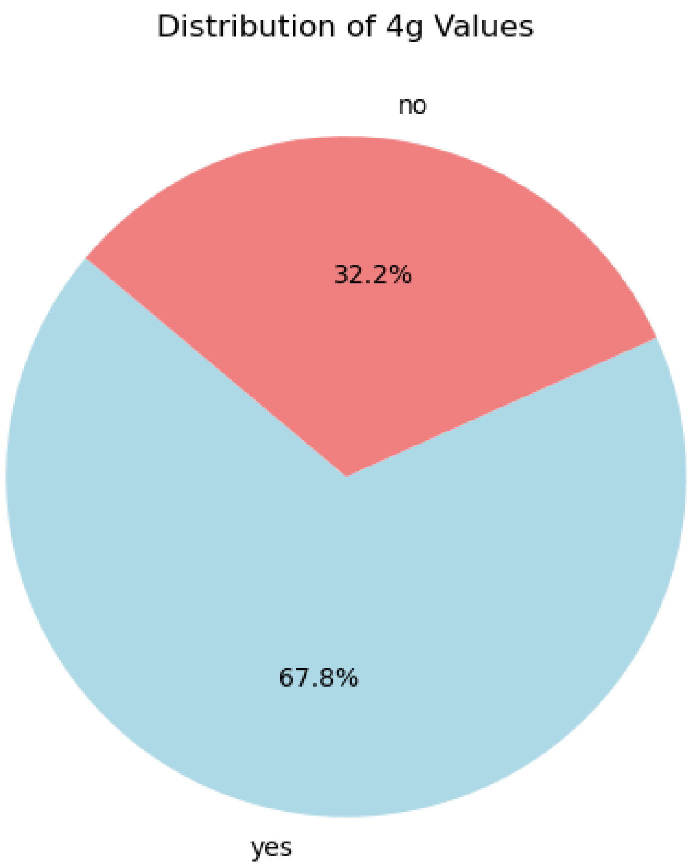
	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
21	Others	Android	20.32	no	no	8.00	0.3	16.0	1.0	5680.0	453.6
57	Huawei	Android	10.16	no	no	5.00	2.0	16.0	4.0	1700.0	136.1

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
58	Huawei	Android	17.78	no	no	3.15	0.3	8.0	1.0	4100.0	350.0
65	Lava	Android	12.70	no	no	5.00	0.3	8.0	0.5	3000.0	147.6
67	Lenovo	Android	25.43	no	no	8.00	5.0	64.0	4.0	7000.0	580.0
...	...	...	...	...	...	...	...	...	...	...	...
3180	ZTE	Android	10.29	no	no	8.00	0.3	16.0	4.0	2000.0	146.0
3181	ZTE	Android	12.70	no	no	8.00	1.3	16.0	4.0	2500.0	163.0
3182	ZTE	Android	10.16	no	no	5.00	0.3	16.0	4.0	1600.0	140.0
3184	ZTE	Android	10.16	no	no	3.15	1.0	16.0	4.0	1600.0	140.0
3185	ZTE	Android	7.75	no	no	3.15	1.0	16.0	4.0	1500.0	140.0

1105 rows × 15 columns

In [114]:

```
count_4g = df['4g'].value_counts()

labels = count_4g.index
sizes = count_4g.values
colors = ['lightblue', 'lightcoral']

plt.figure(figsize=(6, 6))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of 4g Values')
plt.show()
```



In [115]:

```
data_5g = df[df['5g'] == 'yes']
data_5g
```

Out[115]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weigh

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weig
1	Honor	Android	17.20	yes	yes	13.0	16.0	128.0	8.0	4200.0	212.
2	Honor	Android	16.69	yes	yes	13.0	8.0	128.0	8.0	4200.0	213.
3	Honor	Android	25.50	yes	yes	13.0	8.0	64.0	6.0	7250.0	480.
12	Honor	Android	16.69	yes	yes	13.0	16.0	128.0	8.0	4100.0	206.
27	Huawei	Android	15.37	yes	yes	10.5	16.0	128.0	6.0	4000.0	192.
...	...	...	...	...	...	...	...	...	...	...	.
3420	Samsung	Android	15.47	yes	yes	8.0	13.0	128.0	12.0	5000.0	222.
3422	Samsung	Android	15.42	yes	yes	8.0	13.0	128.0	12.0	4500.0	188.
3424	Samsung	Android	15.29	yes	yes	8.0	13.0	128.0	12.0	4000.0	163.
3436	Samsung	Android	17.86	yes	yes	12.0	9.0	512.0	12.0	4235.0	263.
3437	Samsung	Android	15.42	yes	yes	12.0	32.0	128.0	6.0	4500.0	206.

152 rows × 15 columns

In [116]:

```
data_5g1 = df[df['5g'] == 'no']
data_5g1
```

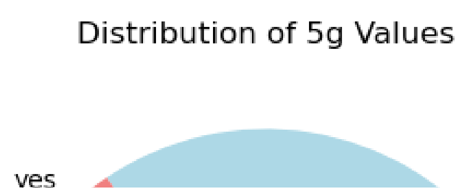
Out[116]:

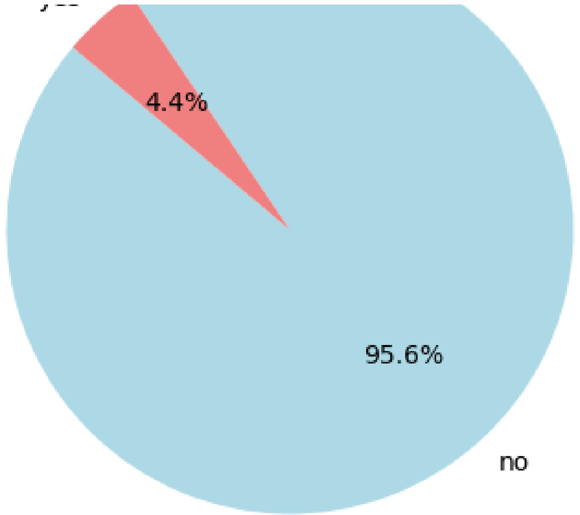
	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
0	Honor	Android	14.50	yes	no	13.000000	5.0	64.0	3.0	3020.0	146.0
4	Honor	Android	15.32	yes	no	13.000000	8.0	64.0	3.0	5000.0	185.0
5	Honor	Android	16.23	yes	no	13.000000	8.0	64.0	4.0	4000.0	176.0
6	Honor	Android	13.84	yes	no	8.000000	5.0	32.0	2.0	3020.0	144.0
7	Honor	Android	15.77	yes	no	13.000000	8.0	64.0	4.0	3400.0	164.0
...	...	...	...	...	...	...	...	...	...	...	...
3449	Asus	Android	15.34	yes	no	9.460208	8.0	64.0	6.0	5000.0	190.0
3450	Asus	Android	15.24	yes	no	13.000000	8.0	128.0	8.0	4000.0	200.0
3451	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	3.0	4000.0	165.0
3452	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	2.0	4000.0	160.0
3453	Alcatel	Android	12.83	yes	no	13.000000	5.0	16.0	2.0	4000.0	168.0

3280 rows × 15 columns

In [117]:

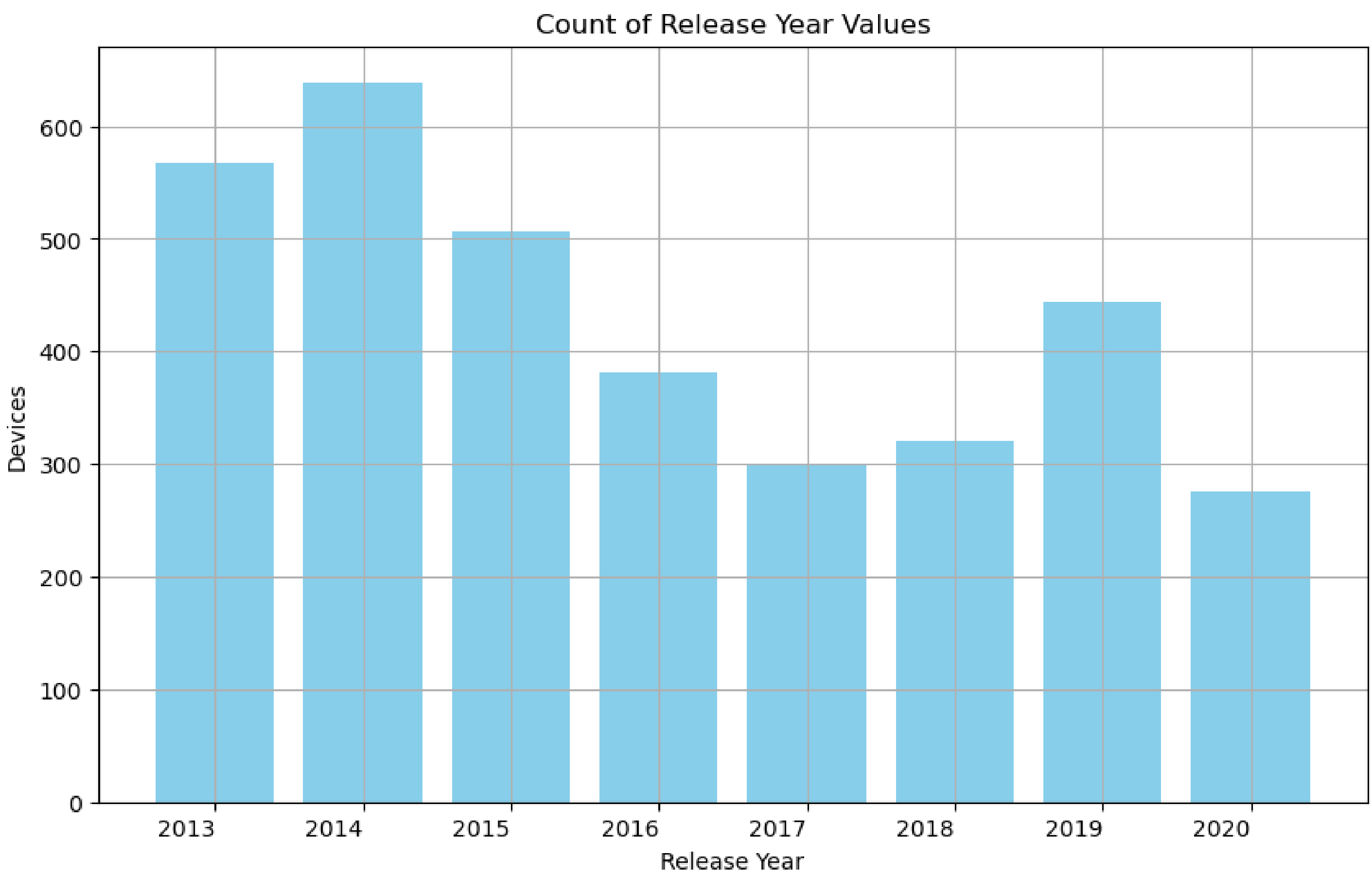
```
count_5g = df['5g'].value_counts()


labels = count_5g.index
sizes = count_5g.values
colors = ['lightblue', 'lightcoral']

plt.figure(figsize=(5, 5))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of 5g Values')
plt.show()
```





In [118]:

```
count_dup = duplicates['release_year'].value_counts()

labels = count_dup.index
counts = count_dup.values

plt.figure(figsize=(10, 6))
plt.bar(labels, counts, color='skyblue')
plt.xlabel('Release Year')
plt.ylabel('Devices')
plt.title('Count of Release Year Values')
plt.xticks(ha='right')
plt.grid()
plt.show()
```



Data Wrangling

In [119]:

```
z_scores = stats.zscore(df['weight'])
print('Z-score is : ',z_scores)
```

```
Z-score is :  0      -0.418656
1        0.342115
2        0.342115
3        3.373845
4        0.024181
           ...
3449     0.080955
3450     0.194502
3451    -0.202915
3452    -0.259689
3453    -0.168851
Name: weight, Length: 3432, dtype: float64
```

In [120]:

```
data_no_outliers = df[(np.abs(z_scores) < 3)]
data_no_outliers
```

Out[120]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weigh
0	Honor	Android	14.50	yes	no	13.000000	5.0	64.0	3.0	3020.0	146.
1	Honor	Android	17.30	yes	yes	13.000000	16.0	128.0	8.0	4300.0	213.
2	Honor	Android	16.69	yes	yes	13.000000	8.0	128.0	8.0	4200.0	213.
4	Honor	Android	15.32	yes	no	13.000000	8.0	64.0	3.0	5000.0	185.
5	Honor	Android	16.23	yes	no	13.000000	8.0	64.0	4.0	4000.0	176.
...	...	...	...	...	...	...	...	...	...	...	.
3449	Asus	Android	15.34	yes	no	9.460208	8.0	64.0	6.0	5000.0	190.
3450	Asus	Android	15.24	yes	no	13.000000	8.0	128.0	8.0	4000.0	200.
3451	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	3.0	4000.0	165.
3452	Alcatel	Android	15.80	yes	no	13.000000	5.0	32.0	2.0	4000.0	160.
3453	Alcatel	Android	12.83	yes	no	13.000000	5.0	16.0	2.0	4000.0	168.

3308 rows × 15 columns

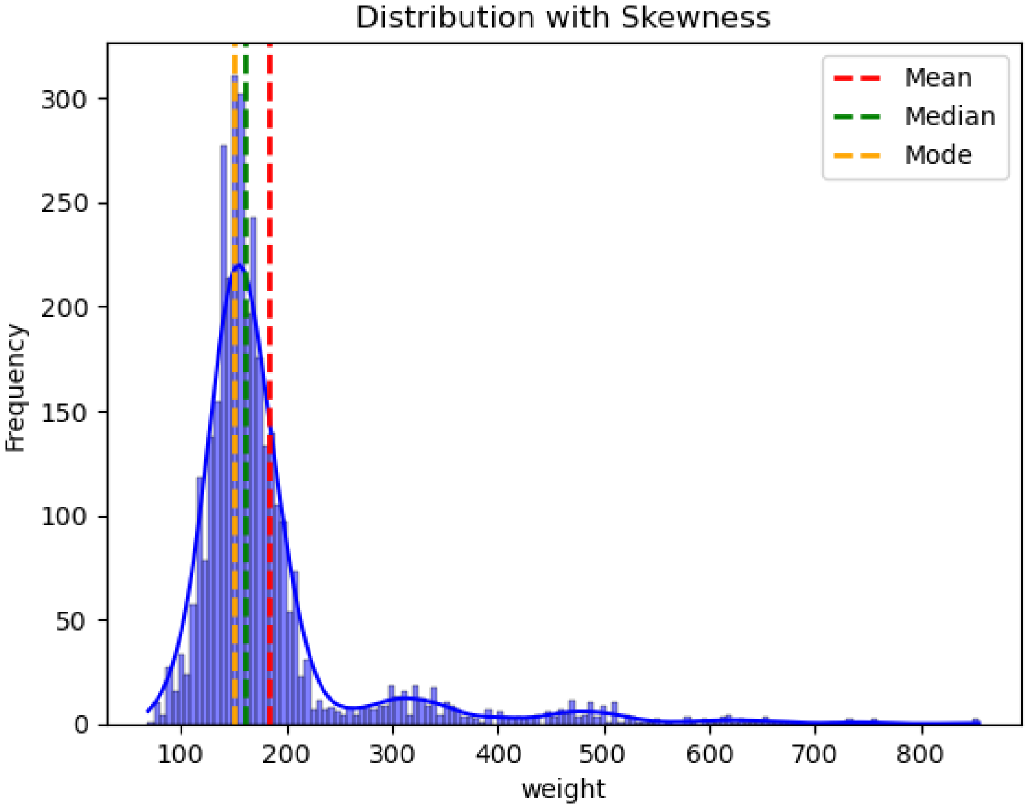
In [121]:

```
skewness_value = skew(df['weight'])
print(f'Skewness: {skewness_value}')
```

```
Skewness: 3.229740230773384
```

In [122]:

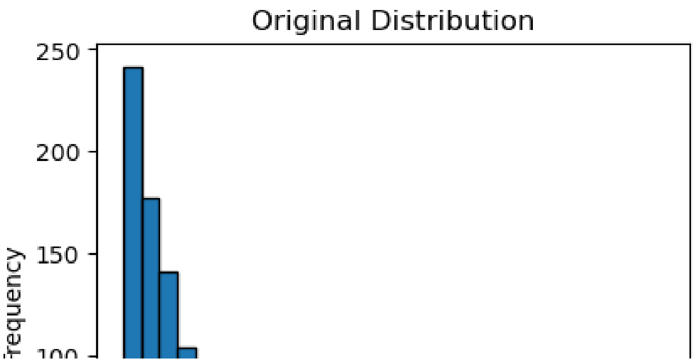
```
sns.histplot(df['weight'], kde=True, color='blue')

plt.axvline(df['weight'].mean(), color='red', linestyle='dashed', linewidth=2, label='Mea
n')
plt.axvline(df['weight'].median(), color='green', linestyle='dashed', linewidth=2, label
='Median')
plt.axvline(df['weight'].mode()[0], color='orange', linestyle='dashed', linewidth=2, lab
el='Mode')

plt.xlabel('weight')
plt.ylabel('Frequency')
plt.title('Distribution with Skewness')
plt.legend()

plt.show()
```

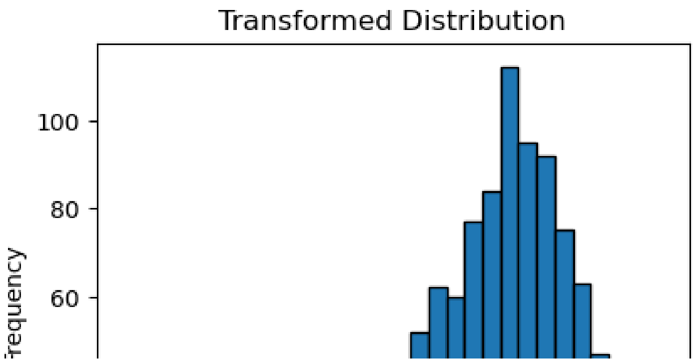


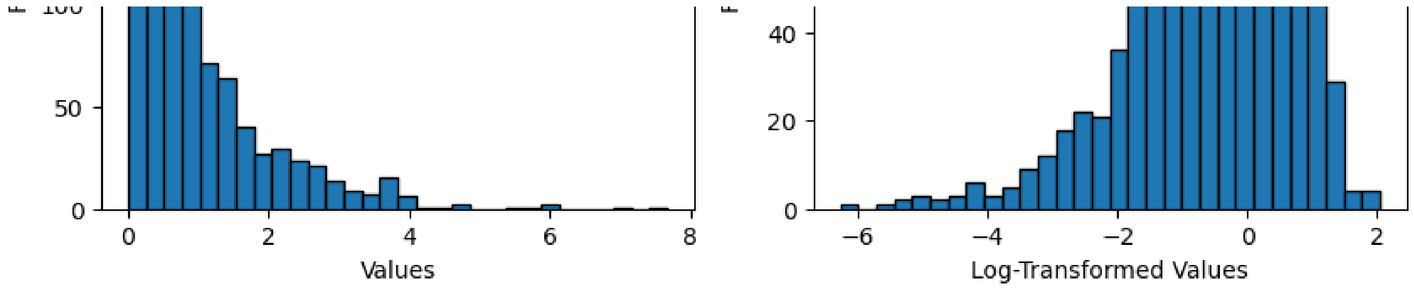
In [123]:

```
data = np.random.exponential(size=1000)

original_skewness = np.mean((data - np.mean(data))**3) / np.std(data)**3
print(f'Original Skewness: {original_skewness}')

transformed_data = np.log(data)

transformed_skewness = np.mean((transformed_data - np.mean(transformed_data))**3) / np.s
td(transformed_data)**3
print(f'Skewness after Log Transformation: {transformed_skewness}')

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)
plt.hist(data, bins=30, edgecolor='black')
plt.title('Original Distribution')
plt.xlabel('Values')
plt.ylabel('Frequency')

plt.subplot(1, 2, 2)
plt.hist(transformed_data, bins=30, edgecolor='black')
plt.title('Transformed Distribution')
plt.xlabel('Log-Transformed Values')
plt.ylabel('Frequency')

plt.show()
```

```
Original Skewness: 2.0044960245717394
Skewness after Log Transformation: -0.9170046820420799
```





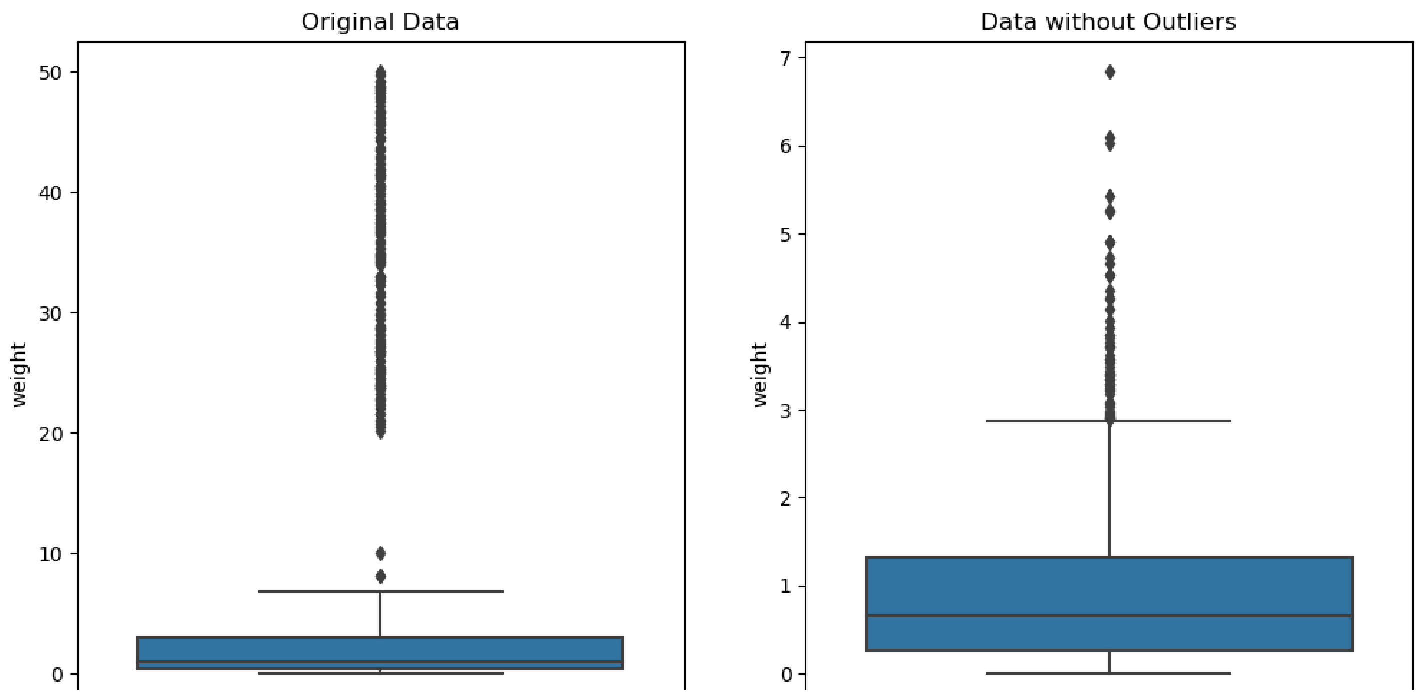


In [124]:

```
Q1 = df['weight'].quantile(0.25)
Q3 = df['weight'].quantile(0.75)

IQR = Q3 - Q1

threshold = 1.5

df_no_outliers = df[(df['weight'] >= (Q1 - threshold * IQR)) & (df['weight'] <= (Q3 + th
reshold * IQR))]
```

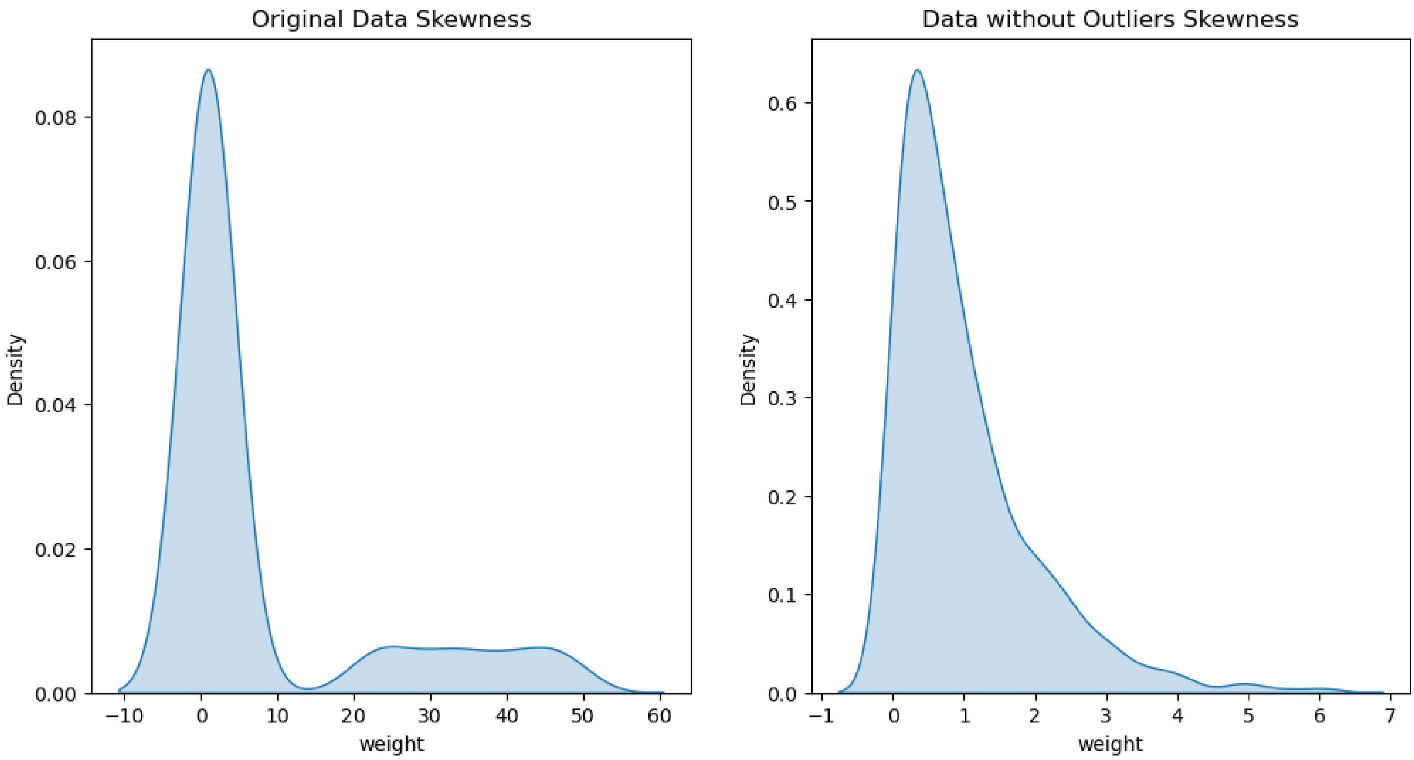
In [126]:

```
data = {'weight': np.concatenate([np.random.exponential(size=800), np.random.uniform(low
=20, high=50, size=200)])}
df = pd.DataFrame(data)

Q1 = df['weight'].quantile(0.25)
Q3 = df['weight'].quantile(0.75)
IQR = Q3 - Q1

threshold = 1.5

df_no_outliers = df[(df['weight'] >= (Q1 - threshold * IQR)) & (df['weight'] <= (Q3 + th
reshold * IQR))]

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.boxplot(y=df['weight'])
plt.title("Original Data")

plt.subplot(1, 2, 2)
sns.boxplot(y=df_no_outliers['weight'])
plt.title("Data without Outliers")

plt.show()
```



In [127]:

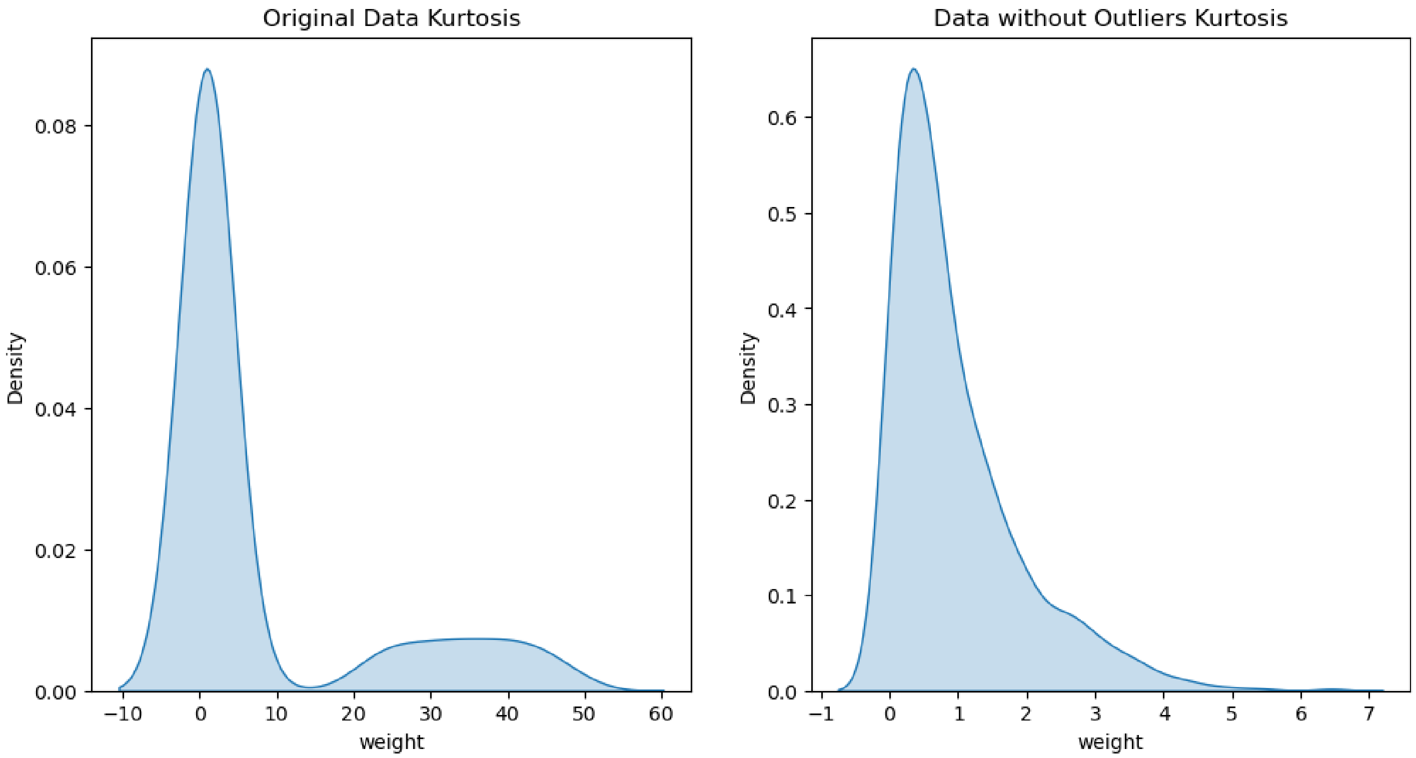
```
data = {'weight': np.concatenate([np.random.exponential(size=800), np.random.uniform(low
=20, high=50, size=200)])}
df = pd.DataFrame(data)

Q1 = df['weight'].quantile(0.25)
Q3 = df['weight'].quantile(0.75)
IQR = Q3 - Q1

threshold = 1.5

df_no_outliers = df[(df['weight'] >= (Q1 - threshold * IQR)) & (df['weight'] <= (Q3 + th
reshold * IQR))]

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.kdeplot(df['weight'], fill=True)
plt.title("Original Data Skewness")

plt.subplot(1, 2, 2)
sns.kdeplot(df_no_outliers['weight'], fill=True)
plt.title("Data without Outliers Skewness")

plt.show()
```



In [128]:

```
data = {'weight': np.concatenate([np.random.exponential(size=800), np.random.uniform(low
=20, high=50, size=200)])}
df = pd.DataFrame(data)

Q1 = df['weight'].quantile(0.25)
Q3 = df['weight'].quantile(0.75)
IQR = Q3 - Q1

threshold = 1.5

df_no_outliers = df[(df['weight'] >= (Q1 - threshold * IQR)) & (df['weight'] <= (Q3 + th
reshold * IQR))]

plt.figure(figsize=(12, 6))
```

```
plt.subplot(1, 2, 1)
sns.kdeplot(df['weight'], fill=True)
plt.title("Original Data Kurtosis")

plt.subplot(1, 2, 2)
sns.kdeplot(df_no_outliers['weight'], fill=True)
plt.title("Data without Outliers Kurtosis")

plt.show()
```

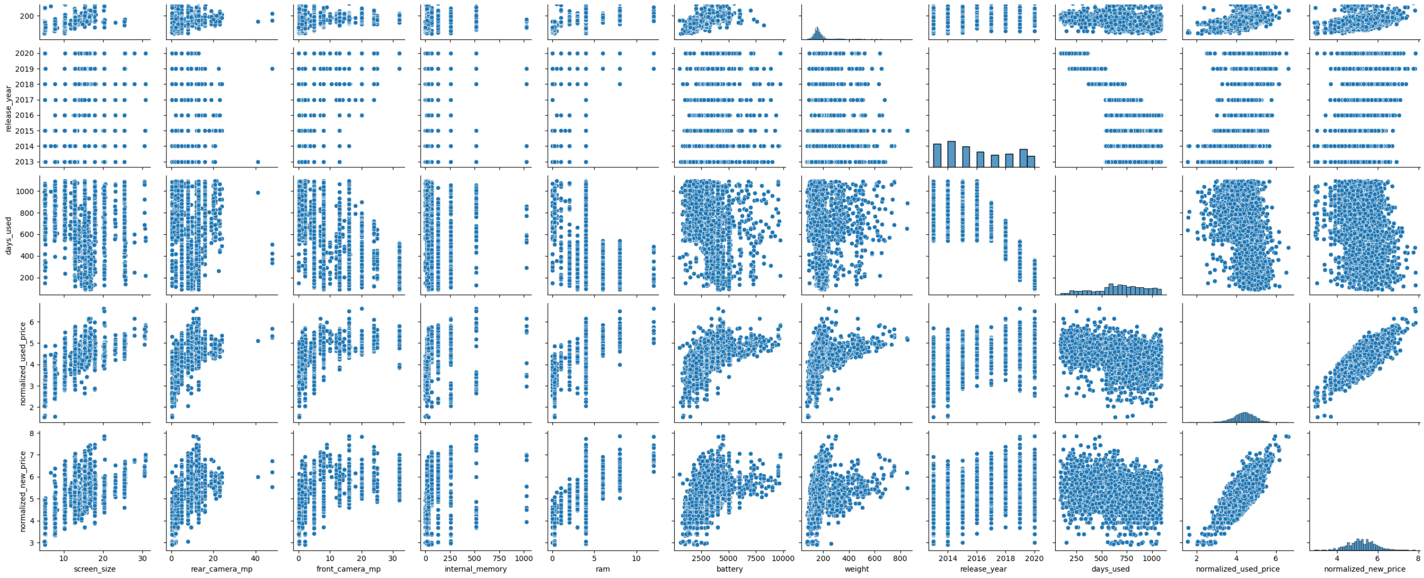


In [140]:

```
sns.pairplot(df)
```

Out[140]:

```
<seaborn.axisgrid.PairGrid at 0x2449e7f5dd0>
```



In [141]:

```
z_scores1 = stats.zscore(df['normalized_used_price'])
print('Z-score is : ',z_scores1)
```

```
Z-score is :  0       -0.104110
1       1.357572
2       1.270313
3       1.311884
4       0.036875
          ...
3449    0.211934
3450    1.144843
3451   -0.018965
3452   -0.031944
3453   -0.404222
Name: normalized_used_price, Length: 3432, dtype: float64
```

In [142]:

```
z_scores2 = stats.zscore(df['normalized_new_price'])
print('Z-score is : ',z_scores2)
```

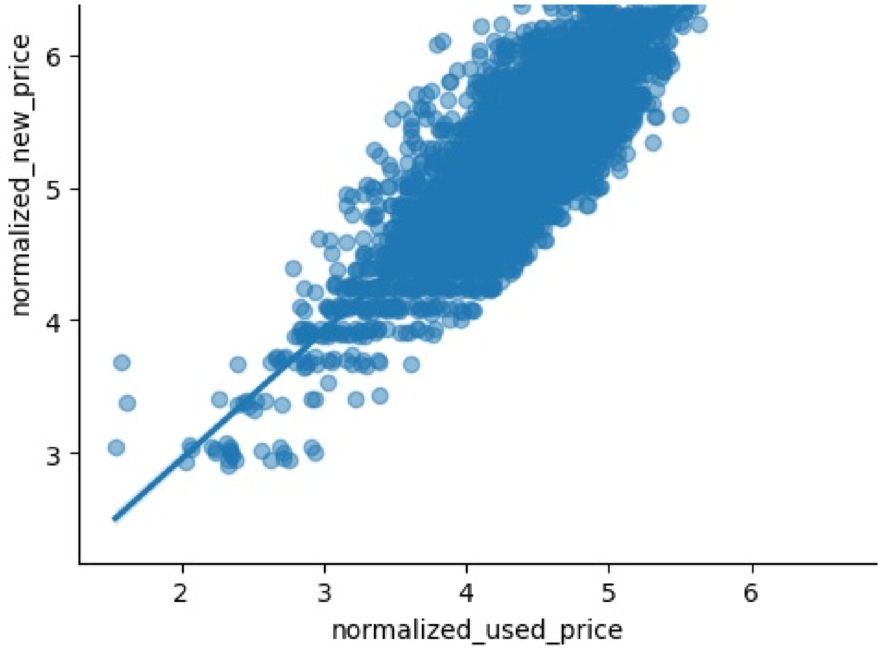
```
Z-score is :  0       -0.769178
1       0.415602
2       0.954426
3       0.580578
4      -0.426180
          ...
3449    1.837562
3450    1.495158
3451   -1.043696
3452   -0.903160
3453   -1.410419
Name: normalized_new_price, Length: 3432, dtype: float64
```

In [143]:

```
sns.lmplot(x = 'normalized_used_price', y = 'normalized_new_price', data = df, scatter_k
ws = {'alpha' : 0.5})
```

Out[143]:

```
<seaborn.axisgrid.FacetGrid at 0x24497007a90>
```



In [144]:

```
df_sorted_asc = df.sort_values(by='normalized_used_price')
df_sorted_asc
```

Out[144]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weig
885	Others	Others	5.08	no	no	0.3	0.3	32.0	0.25	820.0	8
323	Micromax	Android	7.75	no	no	0.3	0.3	0.5	0.25	1500.0	8
533	Alcatel	Others	5.18	no	no	0.3	0.3	16.0	0.25	850.0	7
2320	Others	Others	5.18	no	no	0.3	2.0	64.0	0.25	2100.0	15
2533	Samsung	Others	5.08	no	no	8.0	2.0	16.0	4.00	800.0	7
...	...	...	...	...	...	...	...	...	...	...	
2135	Oppo	Android	16.31	yes	no	13.0	16.0	512.0	4.00	3400.0	18
34	Huawei	Android	16.71	yes	yes	10.5	16.0	256.0	8.00	4200.0	22
645	Apple	iOS	27.94	yes	no	12.0	7.0	1024.0	4.00	7812.0	46
3207	Huawei	Android	20.32	yes	yes	10.5	16.0	512.0	8.00	4500.0	30
198	Xiaomi	Android	20.12	yes	yes	12.0	20.0	512.0	12.00	4050.0	24

3432 rows × 15 columns

In [145]:

```
df_sorted_asc.head(10)
```

Out[145]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
885	Others	Others	5.08	no	no	0.3	0.3	32.0	0.25	820.0	80.0
323	Micromax	Android	7.75	no	no	0.3	0.3	0.5	0.25	1500.0	89.0
533	Alcatel	Others	5.18	no	no	0.3	0.3	16.0	0.25	850.0	77.9
2320	Others	Others	5.18	no	no	0.3	2.0	64.0	0.25	2100.0	150.0
2533	Samsung	Others	5.08	no	no	8.0	2.0	16.0	4.00	800.0	75.0
953	Celkon	Others	5.28	no	no	1.3	0.3	256.0	0.25	1400.0	140.0
884	Others	Others	5.08	no	no	1.3	0.3	128.0	0.25	820.0	80.0

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
898	Others	Others	5.28	[illegible]	[illegible]	1.3	0.3	64.0	0.25	880.0	69.0
954	Celkon	Others	5.23	no	no	1.3	0.3	256.0	0.25	1400.0	140.0
1929	Micromax	Others	5.28	no	no	0.3	0.3	16.0	4.00	2000.0	92.0

In [146]:

```
df_sorted_asc1 = df.sort_values(by='normalized_new_price')
df_sorted_asc1
```

Out[146]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weig
2324	Others	Others	5.18	no	no	0.30	2.0	32.0	0.25	2100.0	15(
2320	Others	Others	5.18	no	no	0.30	2.0	64.0	0.25	2100.0	15(
1904	Micromax	Others	5.16	no	no	0.30	0.3	16.0	4.00	1800.0	11ε
618	Others	Others	5.18	no	no	0.08	2.0	16.0	4.00	1000.0	8(
1903	Micromax	Others	5.18	no	no	0.30	0.3	16.0	4.00	2800.0	26(
...	...	...	...	...	...	...	...	...	...	...	
2358	Samsung	Android	17.86	yes	no	12.00	9.0	512.0	4.00	4380.0	26:
1262	Huawei	Android	20.32	yes	no	13.00	16.0	512.0	4.00	4500.0	29:
198	Xiaomi	Android	20.12	yes	yes	12.00	20.0	512.0	12.00	4050.0	24
3348	Huawei	Android	20.32	yes	yes	10.50	16.0	512.0	8.00	4500.0	30(
3207	Huawei	Android	20.32	yes	yes	10.50	16.0	512.0	8.00	4500.0	30(

3432 rows × 15 columns

In [147]:

```
df_sorted_asc1.head(10)
```

Out[147]:

	device_brand	os	screen_size	4g	5g	rear_camera_mp	front_camera_mp	internal_memory	ram	battery	weight
2324	Others	Others	5.18	no	no	0.30	2.0	32.0	0.25	2100.0	150.0
2320	Others	Others	5.18	no	no	0.30	2.0	64.0	0.25	2100.0	150.0
1904	Micromax	Others	5.16	no	no	0.30	0.3	16.0	4.00	1800.0	118.0
618	Others	Others	5.18	no	no	0.08	2.0	16.0	4.00	1000.0	80.0
1903	Micromax	Others	5.18	no	no	0.30	0.3	16.0	4.00	2800.0	260.0
952	Celkon	Others	5.18	no	no	1.30	0.3	256.0	0.25	1800.0	140.0
1926	Micromax	Others	5.23	no	no	0.30	0.3	16.0	4.00	2000.0	118.0
965	Celkon	Others	5.18	no	no	1.30	0.3	256.0	0.25	1800.0	140.0
1898	Micromax	Others	5.28	no	no	0.30	2.0	16.0	4.00	3000.0	146.5
1924	Micromax	Others	5.28	no	no	0.30	0.3	16.0	4.00	2000.0	108.0

Load or Generate Data

In [34]:

```
cat_cols = ['device_brand','os','4g','5g']
encoder = OneHotEncoder(drop = 'first', sparse = False)
```

```
encoded_cols = pd.DataFrame(encoder.fit_transform(df[cat_cols]), columns = encoder.get_f
eature_names_out(cat_cols))
```

In [35]:

```
cat_cols1 = ['screen_size','rear_camera_mp','front_camera_mp','internal_memory','ram','ba
ttery','weight','release_year',
            'days_used','normalized_used_price']
encoder1 = StandardScaler()
standard_cols = pd.DataFrame(encoder1.fit_transform(df[cat_cols1]), columns = encoder1.ge
t_feature_names_out(cat_cols1))
```

In [36]:

```
x = pd.concat([encoded_cols,standard_cols], axis = 1)
y = df['normalized_new_price']
```

Split Data into Training and Testing Set

In [37]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,train_size = 0.8, random_state =
42) # 42 set at a time
```

In [38]:

```
x_train
```

Out[38]:

	device_brand_Alcatel	device_brand_Apple	device_brand_Asus	device_brand_BlackBerry	device_brand_Celkon	device_bı
3302	0.0	0.0	0.0	0.0	0.0	
2131	0.0	0.0	0.0	0.0	0.0	
572	0.0	0.0	0.0	0.0	0.0	
3124	0.0	0.0	0.0	0.0	0.0	
2713	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	
1095	0.0	0.0	0.0	0.0	0.0	
1130	0.0	0.0	0.0	0.0	0.0	
1294	0.0	0.0	0.0	0.0	0.0	
860	0.0	0.0	0.0	0.0	0.0	
3174	0.0	0.0	0.0	0.0	0.0	

2745 rows × 48 columns

In [39]:

```
y_train
```

Out[39]:

```
3324    5.472229
2146    5.767133
575     5.395898
```

```
3146    5.127529
2728    5.252483
          ...
1100    5.989412
1135    5.297517
1299    5.706844
863     5.563370
3196    5.709566
Name: normalized_new_price, Length: 2745, dtype: float64
```

In [40]:

```
x_test
```

Out[40]:

	device_brand_Alcatel	device_brand_Apple	device_brand_Asus	device_brand_BlackBerry	device_brand_Celkon	device_br
1575	0.0	0.0	0.0	0.0	0.0	
1949	0.0	0.0	0.0	0.0	0.0	
3259	0.0	0.0	0.0	0.0	0.0	
3144	0.0	0.0	0.0	0.0	0.0	
1861	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	
1330	0.0	0.0	0.0	0.0	0.0	
2468	0.0	0.0	0.0	0.0	0.0	
1089	0.0	0.0	0.0	0.0	0.0	
1157	0.0	0.0	0.0	0.0	0.0	
2495	0.0	0.0	0.0	0.0	0.0	

687 rows × 48 columns

In [41]:

```
y_test
```

Out[41]:

```
1580    4.228438
1957    4.489872
3281    5.507281
3166    4.096176
1869    4.938423
          ...
1335    5.299567
2483    5.306335
1094    6.309264
1162    5.348345
2510    5.010835
Name: normalized_new_price, Length: 687, dtype: float64
```

Create a Linear Regression Model

In [42]:

```
model = LinearRegression()
```

Train the Model

In [43]:

```
model.fit(x_train, y_train)
```

Out[43]:

```
▾ LinearRegression
LinearRegression()
```

In [57]:

```
model.coef_
```

Out[57]:

```
array([-0.01804606,  0.7300145 ,  0.00811876,  0.07076426, -0.20602866,
       -0.03133006,  0.04042363,  0.37321618,  0.14524245, -0.16250014,
       -0.00785994, -0.33564817, -0.22055352,  0.1349437 , -0.21990278,
       -0.1012576 ,  0.01932673, -0.19467159, -0.13223842, -0.08311173,
        0.10157468,  0.21736106,  0.06929888,  0.02529324, -0.09090754,
       -0.22424105,  0.14982965,  0.06174723, -0.24358829,  0.03292971,
       -0.05942954, -0.18624477, -0.08788289, -0.18796876, -0.00667928,
       -0.06258604,  0.12423129,  0.26774169,  0.03239445,  0.06916135,
        0.01489565,  0.05269081,  0.07192952,  0.04727583, -0.06530555,
       -0.20345336, -0.02138842,  0.45829171])
```

In [59]:

```
cdf = pd.DataFrame(model.coef_, x.columns, columns = ['coef'])
cdf
```

Out[59]:

	coef
device_brand_Alcatel	-0.018046
device_brand_Apple	0.730014
device_brand_Asus	0.008119
device_brand_BlackBerry	0.070764
device_brand_Celkon	-0.206029
device_brand_Coolpad	-0.031330
device_brand_Gionee	0.040424
device_brand_Google	0.373216
device_brand_HTC	0.145242
device_brand_Honor	-0.162500
device_brand_Huawei	-0.007860
device_brand_Infinix	-0.335648
device_brand_Karbonn	-0.220554
device_brand_LG	0.134944
device_brand_Lava	-0.219903
device_brand_Lenovo	-0.101258
device_brand_Meizu	0.019327
device_brand_Micromax	-0.194672
device_brand_Microsoft	-0.132238

	coef
device_brand_Motorola	-0.083112
device_brand_Nokia	0.101575
device_brand_OnePlus	0.217361
device_brand_Oppo	0.069299
device_brand_Others	0.025293
device_brand_Panasonic	-0.090908
device_brand_Realme	-0.224241
device_brand_Samsung	0.149830
device_brand_Sony	0.061747
device_brand_Spice	-0.243588
device_brand_Vivo	0.032930
device_brand_XOLO	-0.059430
device_brand_Xiaomi	-0.186245
device_brand_ZTE	-0.087883
os_Others	-0.187969
os_Windows	-0.006679
os_iOS	-0.062586
4g_yes	0.124231
5g_yes	0.267742
screen_size	0.032394
rear_camera_mp	0.069161
front_camera_mp	0.014896
internal_memory	0.052691
ram	0.071930
battery	0.047276
weight	-0.065306
release_year	-0.203453
days_used	-0.021388
normalized_used_price	0.458292

Make Predictions

In [44]:

```
y_pred = model.predict(x_test)
```

Actual and Predicted Data

In [45]:

```
a = {'actual':y_test, 'prediction':y_pred}
pd.DataFrame(data=a)
```

Out[45]:

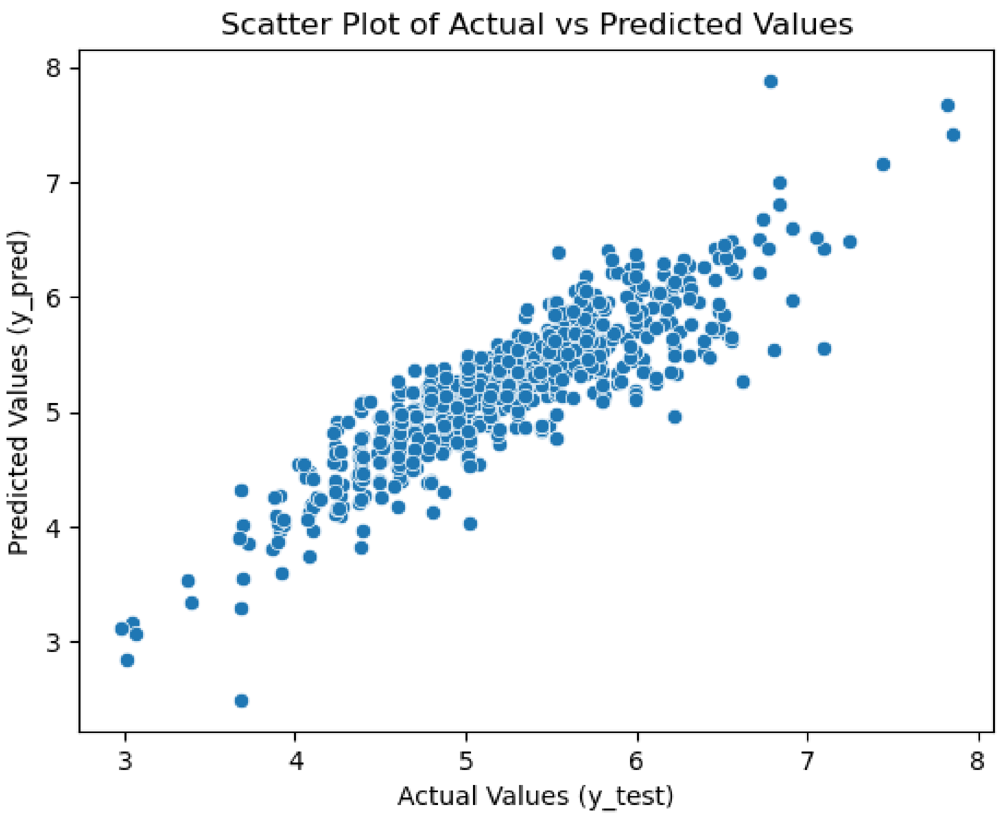
	actual	prediction
1580	4.228438	4.363960
1957	4.489872	4.396655

	actual	prediction
3281	5.507281	5.939829
3166	4.096176	4.198515
1869	4.938423	4.926316
...	...	...
1335	5.299567	5.535421
2483	5.306335	5.351722
1094	6.309264	5.484360
1162	5.348345	5.655820
2510	5.010835	4.830719

687 rows × 2 columns

In [64]:

```
sns.scatterplot(x=y_test, y=y_pred)
plt.xlabel('Actual Values (y_test)')
plt.ylabel('Predicted Values (y_pred)')
plt.title('Scatter Plot of Actual vs Predicted Values')
plt.show()
```



In [48]:

```
r2 = r2_score(y_test, y_pred)
print('R2 Score : ',r2)
```

```
R2 Score :  0.7731314964499362
```

In [49]:

```
m_a_e = mean_absolute_error(y_test,y_pred)
print('Mean Absolute Error : ',m_a_e)
```

```
Mean Absolute Error :  0.25394503963029696
```

In [50]:

```
m_s_e = mean_squared_error(y_test,y_pred)
print('Mean Squared Error : ',m_s_e)
```

```
Mean Squared Error :  0.11215986262530579
```

In [51]:

```python
RMSE = np.sqrt(m_s_e)
print('Root Mean Square Error :', RMSE)
```

```
Root Mean Square Error : 0.334902765926628
```

Residuals

In [66]:

```python
residuals = y_test - y_pred
residuals
```

Out[66]:

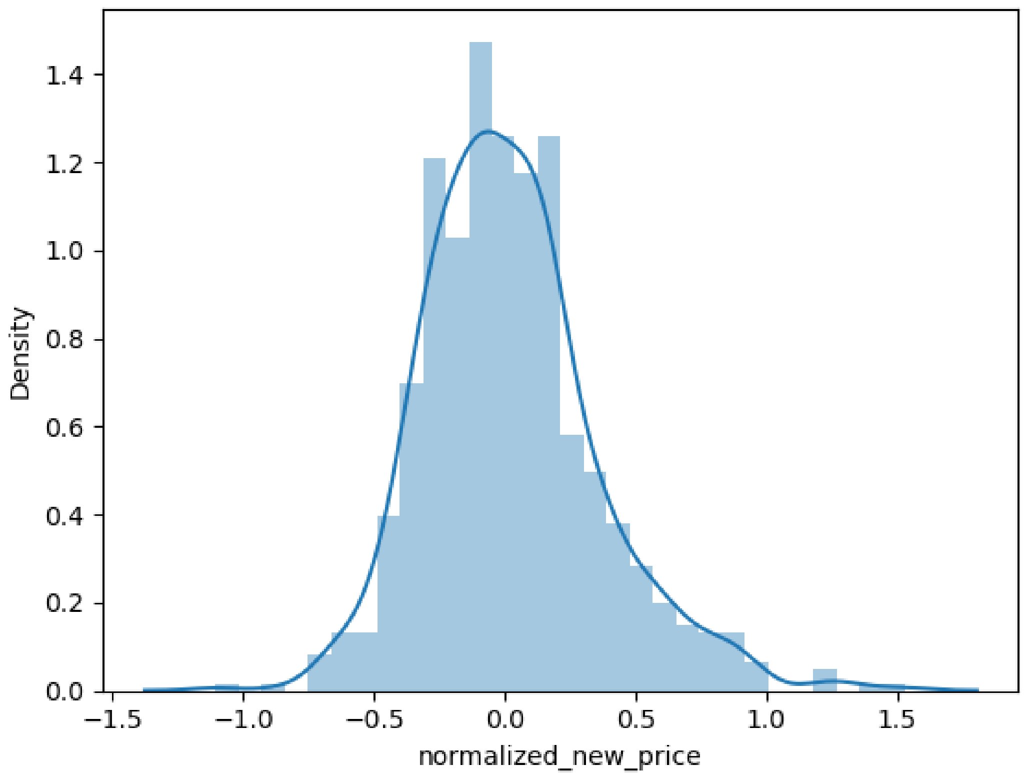
```
1580   -0.135521
1957    0.093217
3281   -0.432548
3166   -0.102339
1869    0.012107
          ...
1335   -0.235854
2483   -0.045387
1094    0.824903
1162   -0.307475
2510    0.180117
Name: normalized_new_price, Length: 687, dtype: float64
```

In [67]:

```python
sns.distplot(residuals)
```

Out[67]:

```
<Axes: xlabel='normalized_new_price', ylabel='Density'>
```



Hyperparameter Tuning

In [54]:

```
initial_mse = mean_squared_error(y_test, y_pred)
print("Initial Mean Squared Error:", initial_mse)

param_grid = {
    'fit_intercept': [True, False]
}

grid_search = GridSearchCV(estimator=LinearRegression(), param_grid=param_grid, scoring=
'neg_mean_squared_error', cv=5)

grid_search.fit(x_train, y_train)

best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

best_model = grid_search.best_estimator_
y_pred_tuned = best_model.predict(x_test)

tuned_mse = mean_squared_error(y_test, y_pred_tuned)
print("Tuned Mean Squared Error:", tuned_mse)
```

```
Initial Mean Squared Error: 0.11215986262530579
Best Hyperparameters: {'fit_intercept': True}
Tuned Mean Squared Error: 0.11215986262530579
```

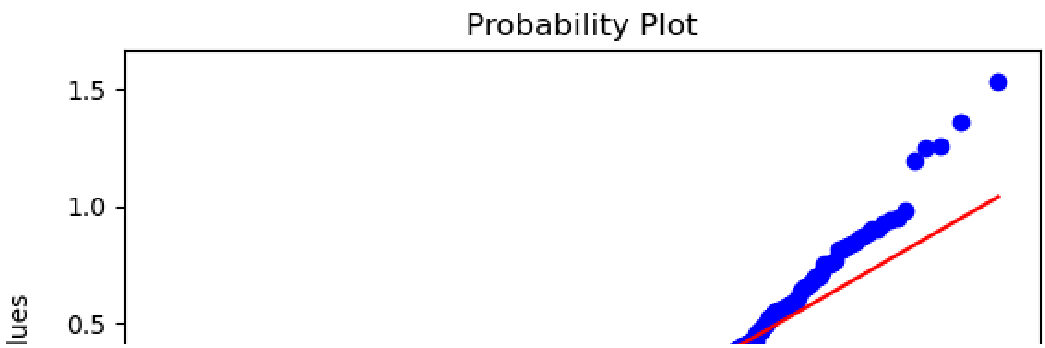
In [55]:

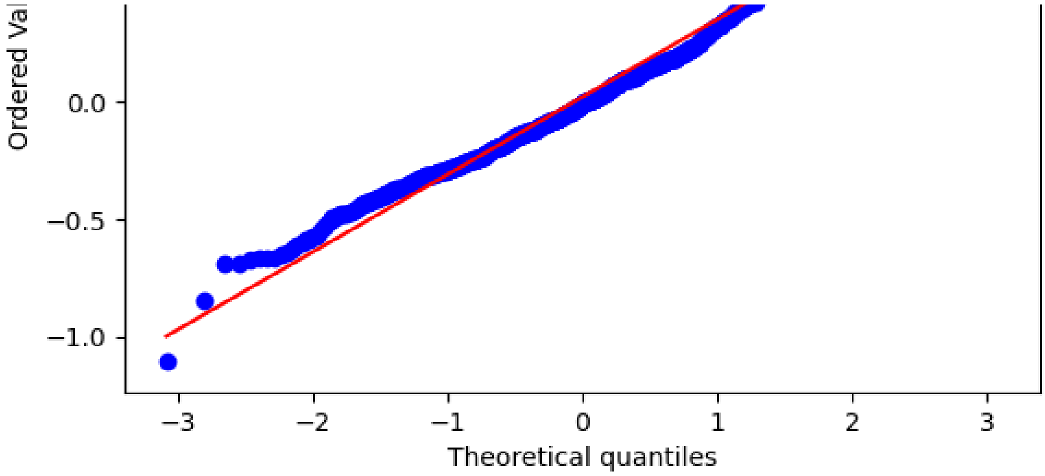
```
param_dist = {
    'fit_intercept': [True, False],
}
random_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_dist,
    n_iter=10,
    scoring='neg_mean_squared_error',
    cv=5,
    random_state=42
)

random_search.fit(x_train, y_train)

best_params = random_search.best_params_
print("Best Hyperparameters:", best_params)

best_model = random_search.best_estimator_
y_pred_tuned = best_model.predict(x_test)

tuned_mse = mean_squared_error(y_test, y_pred_tuned)
print("Tuned Mean Squared Error:", tuned_mse)
```

```
Best Hyperparameters: {'fit_intercept': True}
Tuned Mean Squared Error: 0.11215986262530579
```

In [68]:

```
import pylab
import scipy.stats as stats
stats.probplot(residuals, dist = 'norm', plot = pylab)
pylab.show()
```





In [ ]: