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

In [2]:
```
df=pd.read_csv("Assignment - Junior Data Analyst (1).csv")
```

In [3]:
```
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = df.select_dtypes(include=['object', 'category']).columns
```

```
In [4]: df.head(10)
```

Out[4]:

	battery	camera	display	memory	name	price	processor	rating	reviews	warra
0	5000 mAh Battery	12MP + 2MP | 8MP Front Camera	15.8 cm (6.22 inch) HD+ Display	4 GB RAM | 64 GB ROM | Expandable Upto 512 GB	Redmi 8 (Ruby Red, 64 GB)	9999	Qualcomm Snapdragon 439 Processor	4.4	55,078 Reviews	Br Warra of 1 Y Availa Mobil
1	5000 mAh Battery	12MP + 8MP + 2MP + 2MP | 8MP Front Camera	16.56 cm (6.52 inch) HD+ Display	4 GB RAM | 64 GB ROM	Realme 5i (Aqua Blue, 64 GB)	10999	Qualcomm Snapdragon 665 2 GHz Processor	4.5	20,062 Reviews	Sun Des
2	5000 mAh Battery	12MP + 8MP + 2MP + 2MP | 8MP Front Camera	16.56 cm (6.52 inch) HD+ Display	4 GB RAM | 128 GB ROM	Realme 5i (Aqua Blue, 128 GB)	11999	Qualcomm Snapdragon 665 (2 GHz) Processor	4.5	20,062 Reviews	Sun Des
3	5000 mAh Battery	12MP + 8MP + 2MP + 2MP | 8MP Front Camera	16.56 cm (6.52 inch) HD+ Display	4 GB RAM | 128 GB ROM	Realme 5i (Forest Green, 128 GB)	11999	Qualcomm Snapdragon 665 (2 GHz) Processor	4.5	20,062 Reviews	Sun Des
4	4000 mAh Battery	13MP + 2MP | 5MP Front Camera	15.49 cm (6.1 inch) HD+ Display	3 GB RAM | 32 GB ROM | Expandable Upto 256 GB	Realme C2 (Diamond Blue, 32 GB)	7499	MediaTek P22 Octa Core 2.0 GHz Processor	4.4	10,091 Reviews	D N SIM s Mem Card
5	4000 mAh Battery	13MP + 2MP | 5MP Front Camera	15.49 cm (6.1 inch) HD+ Display	3 GB RAM | 32 GB ROM | Expandable Upto 256 GB	Realme C2 (Diamond Black, 32 GB)	7499	MediaTek P22 Octa Core 2.0 GHz Processor	4.4	10,091 Reviews	D N SIM s Mem Card
6	4000 mAh Battery	13MP + 2MP | 5MP Front Camera	15.49 cm (6.1 inch) HD+ Display	2 GB RAM | 32 GB ROM | Expandable Upto 256 GB	Realme C2 (Diamond Black, 32 GB)	6999	MediaTek P22 Octa Core 2.0 GHz Processor	4.4	67,674 Reviews	D N SIM s Mem Card
7	4000 mAh Battery	13MP + 2MP | 5MP Front Camera	15.49 cm (6.1 inch) HD+ Display	3 GB RAM | 32 GB ROM | Expandable Upto 256 GB	Realme C2 (Diamond Sapphire, 32 GB)	7499	MediaTek P22 Octa Core 2.0 GHz Processor	4.4	10,091 Reviews	D N SIM s Mem Card
8	5000 mAh Battery	12MP + 8MP + 2MP + 2MP | 8MP Front Camera	16.56 cm (6.52 inch) HD+ Display	4 GB RAM | 64 GB ROM	Realme 5i (Forest Green, 64 GB)	10999	Qualcomm Snapdragon 665 2 GHz Processor	4.5	20,062 Reviews	Sun Des

	battery	camera	display	memory	name	price	processor	rating	reviews	warra
9	4000 mAh Battery	13MP + 2MP | 5MP Front Camera	15.49 cm (6.1 inch) HD+ Display	3 GB RAM | 32 GB ROM | Expandable Upto 256 GB	Realme C2 (Diamond Ruby, 32 GB)	7499	MediaTek P22 Octa Core 2.0 GHz Processor	4.4	10,091 Reviews	D Na SIM s a Mem Card

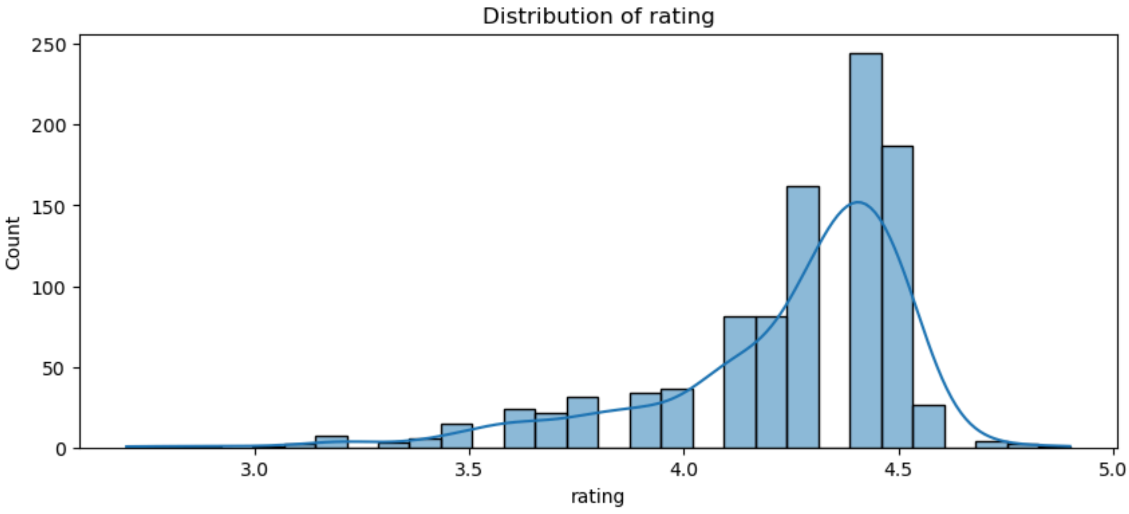
In [5]:

```
for col in numerical_cols:
    print(f"Univariate analysis of numerical column: {col}")
    print(df[col].describe())
```

```
Univariate analysis of numerical column: price
count       984.000000
mean      15429.848577
std       12891.355967
min         887.000000
25%        7499.000000
50%       11649.000000
75%       17999.250000
max      104999.000000
Name: price, dtype: float64
Univariate analysis of numerical column: rating
count    971.000000
mean       4.241195
std        0.300296
min        2.700000
25%        4.100000
50%        4.300000
75%        4.400000
max        4.900000
Name: rating, dtype: float64
```

In [6]:

```
plt.figure(figsize=(10, 4))
sns.histplot(df[col], kde=True, bins=30)
plt.title(f'Distribution of {col}')
plt.show()
```



In [8]:

```
print(df[col].dtype)
print(df[col].head())
```

```
float64
0    4.4
1    4.5
2    4.5
3    4.5
4    4.4
Name: rating, dtype: float64
```

In [9]:

```
df['battery'] = df['battery'].str.extract('(\d+)').astype(float)
```

In [10]:

```
correlation_matrix = df.select_dtypes(include=[float, int]).corr()
```

In [11]:

```
correlation_matrix = df.apply(pd.to_numeric, errors='coerce').corr()
```

In [12]:

```
correlation_matrix = df.apply(pd.to_numeric, errors='coerce').corr()

```

In [ ]:

```

df_numeric = df.select_dtypes(include=[float, int])

correlation_matrix = df_numeric.corr()

print(correlation_matrix)

```

In [ ]:

```

print(df.dtypes)
df['column_name'] = pd.to_numeric(df['column_name'], errors='coerce')

```

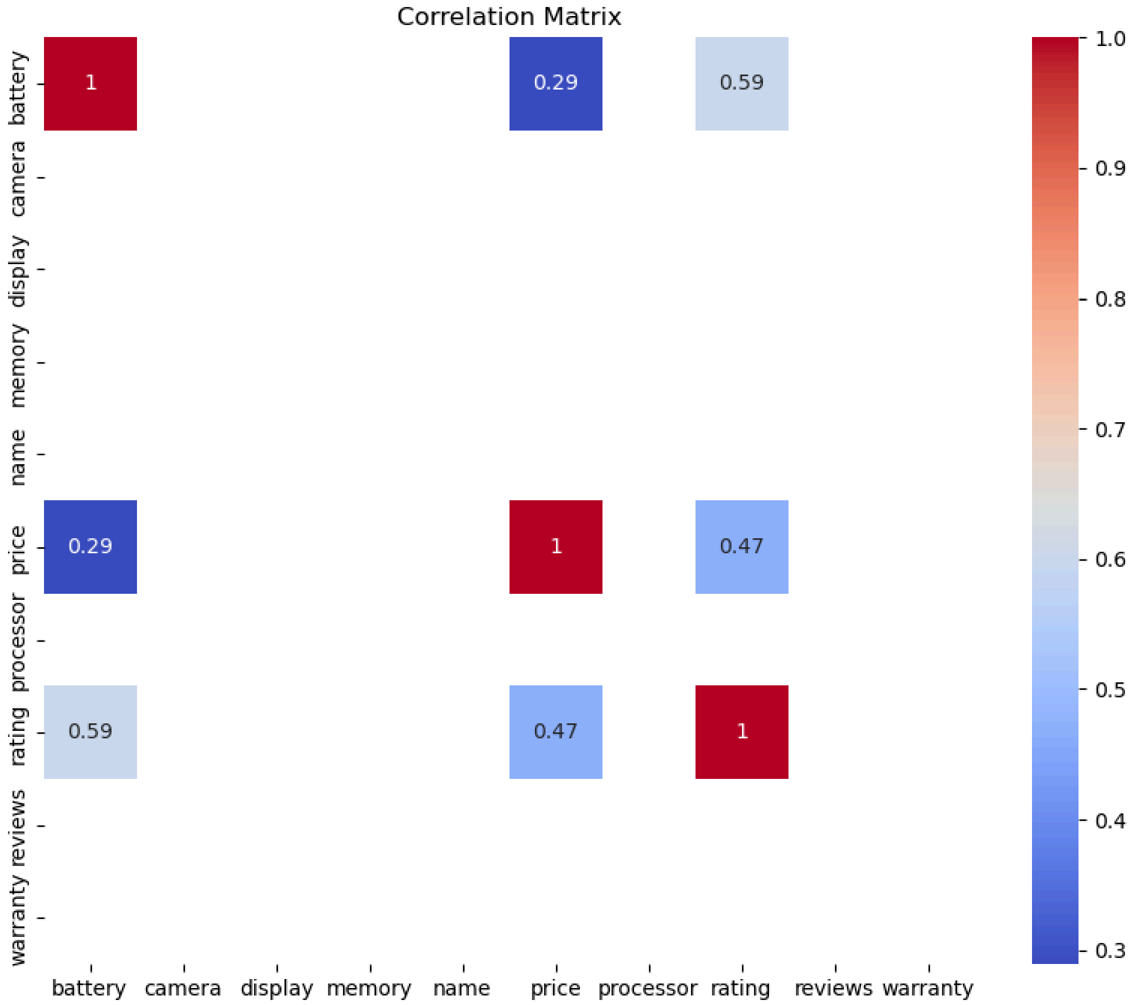
In [30]:

```

df_numeric = df_numeric.dropna()
```

In [17]:

```
correlation_matrix = df.apply(pd.to_numeric, errors='coerce').corr()

price_corr = correlation_matrix['price'].sort_values(ascending=False)
rating_corr = correlation_matrix['rating'].sort_values(ascending=False)

print("Correlation with Price:")
print(price_corr)

print("\nCorrelation with Rating:")
print(rating_corr)

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

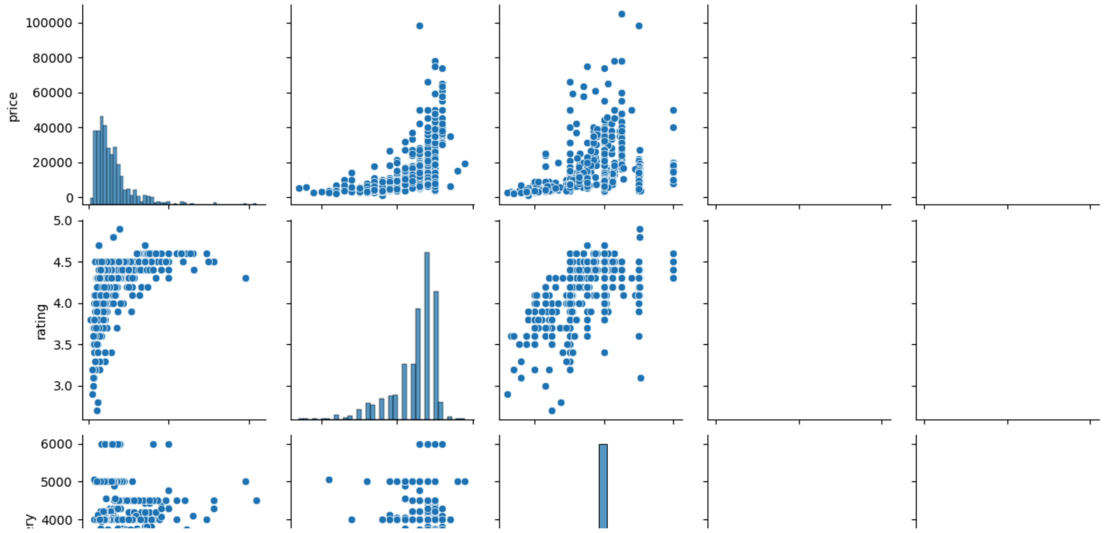
```
Correlation with Price:
price        1.000000
rating       0.469329
battery      0.288851
camera            NaN
display           NaN
memory            NaN
name              NaN
processor         NaN
reviews           NaN
warranty          NaN
Name: price, dtype: float64

Correlation with Rating:
rating       1.000000
battery      0.594831
price        0.469329
camera            NaN
display           NaN
memory            NaN
name              NaN
processor         NaN
reviews           NaN
warranty          NaN
Name: rating, dtype: float64
```



In [18]:

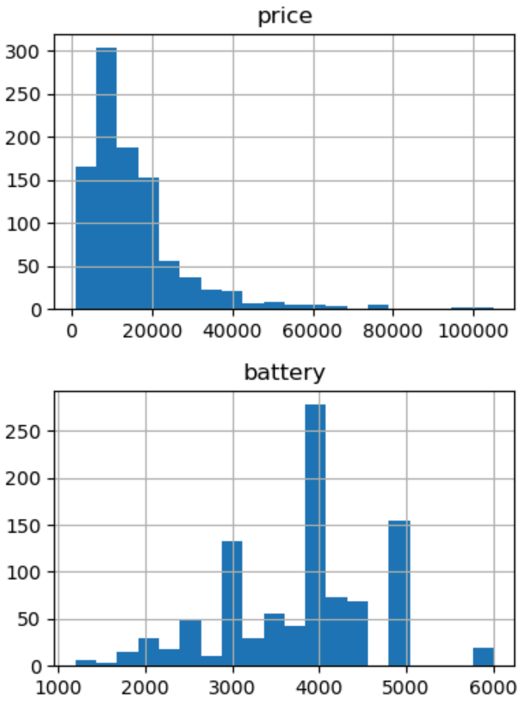
```
# other insights that can we drawn are
sns.pairplot(df[['price', 'rating', 'battery', 'memory', 'camera']].apply(p
plt.show()
```

```
C:\Users\lenovo\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: Use
rWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



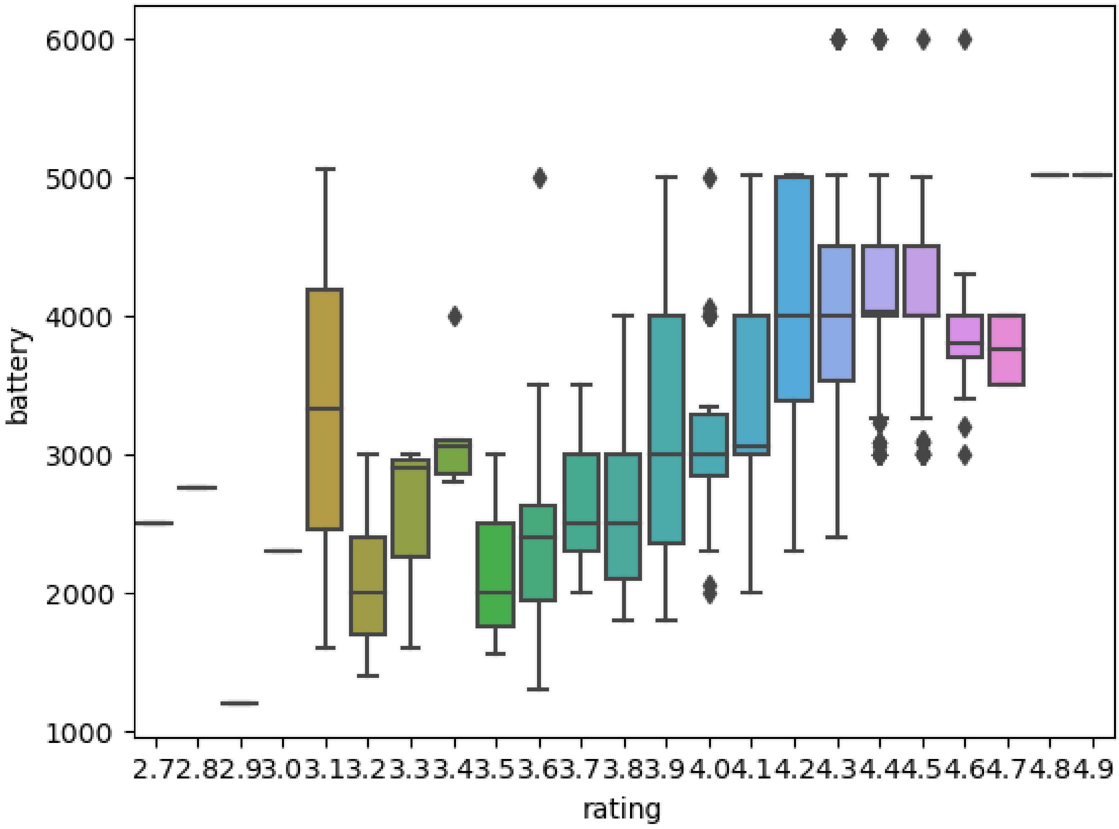
In [19]:

```
# Distribution of Features
df[['price', 'rating', 'battery']].hist(bins=20, figsize=(10, 6))
plt.show()
```



In [20]:

```
# Relationship between Features and Ratings
sns.boxplot(x='rating', y='battery', data=df)
plt.show()
```



In [22]:

```
# Average Price and Rating by Feature
avg_price_rating = df.groupby('processor')[['price', 'rating']].mean().sort_
print(avg_price_rating)
```

```
                                                      price    rating
processor                                                            
Exynos Octa Core Processor                          95999.0       NaN
Exynos 990 Processor                                85979.0  4.420000
Snapdragon 845 Octa Core 2.649 GHz Processor        60990.0  4.600000
Exynos 9 9820 Processor                             60000.0  4.571429
Qualcomm Snapdragon™ 855 Octa-core (up to 2.84 ...  54990.0  4.600000
...                                                     ...       ...
Mediatek Processor                                   3049.0  3.800000
MTK Processor                                        3015.5  3.600000
Spreadtrum SC7731C Quad Core 1.2GHz Processor        2999.0  3.550000
Quad Core, SC9850K Processor Processor               2850.0  3.700000
Spreadtrum SPR7715 Processor                         1999.0  3.200000

[287 rows x 2 columns]
```

In [27]:

```
df['battery'] = df['battery'].astype(str).str.extract('(\d+)').astype(float
df['camera'] = df['camera'].astype(str).str.extract('(\d+)').astype(float)
df['warranty'] = df['warranty'].astype(str).str.extract('(\d+)').astype(flo
df['price'] = pd.to_numeric(df['price'], errors='coerce')  # Convert price 

df = df.fillna(df.mean())

print(f"Shape of dataset after preprocessing: {df.shape}")

X = df.drop(columns=['rating'])
y = df['rating']

print(f"Shape of X: {X.shape}")
print(f"Shape of y: {y.shape}")

if X.shape[0] > 0 and y.shape[0] > 0:

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2


    model = LinearRegression()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"Mean Squared Error: {mse}")
    print(f"R^2 Score: {r2}")

    plt.scatter(y_test, y_pred)
    plt.xlabel("Actual Ratings")
    plt.ylabel("Predicted Ratings")
    plt.title("Actual vs Predicted Ratings")
    plt.show()
else:
    print("Error: No data available for model training.")
```

```
Shape of dataset after preprocessing: (0, 10)
Shape of X: (0, 9)
Shape of y: (0,)
Error: No data available for model training.
```

In [26]:

```

df['battery'] = df['battery'].astype(str).str.extract('(\d+)').astype(float
df['camera'] = df['camera'].astype(str).str.extract('(\d+)').astype(float)
df['warranty'] = df['warranty'].astype(str).str.extract('(\d+)').astype(floa
df['price'] = pd.to_numeric(df['price'], errors='coerce')

df_numeric = df.drop(columns=['name', 'reviews', 'rating'])

print("Before filling missing values:")
print(df_numeric.isna().sum())

df_numeric = df_numeric.fillna(df_numeric.mean())

print("After filling missing values:")
print(df_numeric.isna().sum())


print("Shape of the dataset after preprocessing:", df_numeric.shape)

if df_numeric.shape[0] == 0:
    print("Error: No valid samples left for clustering.")
else:

    scaler = StandardScaler()
    df_scaled = scaler.fit_transform(df_numeric)

    kmeans = KMeans(n_clusters=4, random_state=42)
    df['cluster'] = kmeans.fit_predict(df_scaled)

    pca = PCA(n_components=2)
    df_pca = pca.fit_transform(df_scaled)

    plt.figure(figsize=(8, 6))
    plt.scatter(df_pca[:, 0], df_pca[:, 1], c=df['cluster'], cmap='viridis'
    plt.title('K-Means Clustering of Phones (PCA-reduced)')
    plt.xlabel('PCA 1')
    plt.ylabel('PCA 2')
    plt.colorbar(label='Cluster')
    plt.show()
```

```
Before filling missing values:
battery      0.0
camera       0.0
display      0.0
memory       0.0
price        0.0
processor    0.0
warranty     0.0
dtype: float64
After filling missing values:
battery      0.0
camera       0.0
display      0.0
memory       0.0
price        0.0
processor    0.0
warranty     0.0
dtype: float64
Shape of the dataset after preprocessing: (0, 7)
Error: No valid samples left for clustering.
```

In [ ]: