```python
import pandas as pd
import numpy as np

df = pd.read_csv('smartphoneamazon.csv')
# Remove leading/trailing spaces from column headers
df.columns = df.columns.str.strip()


def clean_price(x):
    if pd.isna(x):
        return np.nan
    x = str(x).replace("₹", "").replace(",", "").strip()
    return pd.to_numeric(x, errors='coerce')

df['listed_price'] = df['listed_price'].apply(clean_price)
df['mrp'] = df['mrp'].apply(clean_price)

def clean_reviews(x):
    if pd.isna(x):
        return np.nan
    x = str(x).replace("(", "").replace(")", "").strip()
    if "K" in x:
        return float(x.replace("K", "")) * 1000
    x = pd.to_numeric(x, errors='coerce')
    if x < 0:
        return np.nan
    return x

df['review_count'] = df['review_count'].apply(clean_reviews)

df['rating'] = pd.to_numeric(df['rating'], errors='coerce')

def extract_discount(x):
    if pd.isna(x):
        return np.nan
    x = str(x)
    if "%" in x:
        return float(x.split("%")[0].replace("(", "").strip())
    return np.nan

df['discount_percent'] = df['discount'].apply(extract_discount)

df_clean = df[
    (df['listed_price'] > 0) &
    (df['mrp'] > df['listed_price']) &
    (df['rating'] >= 1) & (df['rating'] <= 5) &
    (df['review_count'] >= 0)
]

df_clean[features].isna().sum()
```

```
listed_price        0
mrp                 0
rating              0
review_count        0
discount_percent    0
price_diff          0
dtype: int64
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_clean[features])
```

```
from sklearn.ensemble import IsolationForest

iso = IsolationForest(
    n_estimators=200,
    contamination=0.1,
    random_state=42
)

iso.fit(X_scaled)
```

```
IsolationForest(contamination=0.1, n_estimators=200, random_state=42)
```

```
df_clean['anomaly_score'] = iso.decision_function(X_scaled)
df_clean['anomaly_flag'] = iso.predict(X_scaled)
```

```
/tmp/ipython-input-2927649378.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_clean['anomaly_score'] = iso.decision_function(X_scaled)
/tmp/ipython-input-2927649378.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_clean['anomaly_flag'] = iso.predict(X_scaled)
```

```
df_clean.head()
```


```
{"summary":"{\n  \"name\": \"df_clean\",\n  \"rows\": 133,\n
\"fields\": [\n    {\n      \"column\": \"listed_price\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
```

```
12473.334775359983,\n        \"min\": 79.0,\n        \"max\":
70999.0,\n        \"num_unique_values\": 61,\n        \"samples\": [\n
26999.0,\n          9999.0,\n          109.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"delivery_date\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 6,\n        \"samples\": [\n
\"Tomorrow, 19 Dec\",\n          \"Sat, 20 Dec\",\n          \"20 - 26
Dec\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"product_name\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 82,\n        \"samples\":
[\n          \"OnePlus 13R | Smarter with OnePlus AI | Lifetime
Display Warranty (12GB RAM, 256GB Storage Nebula Noir)\",\n
\"Samsung Galaxy A55 5G (Awesome Iceblue, 8GB RAM, 256GB Storage) | AI
| Metal Frame | 50 MP Main Camera (OIS) | Super HDR Video|
Nightography | IP67 | Corning Gorilla Glass Victus+ | sAMOLED
Display\",\n          \"Vivo T4x 5G (Marine Blue, 128 GB) (8 GB
RAM)\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"review_count\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 5452.70547875545,\n        \"min\":
1000.0,\n        \"max\": 34600.0,\n        \"num_unique_values\":
42,\n        \"samples\": [\n          2900.0,\n          1700.0,\n
3500.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"mrp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 17498.784061441493,\n        \"min\": 299.0,\n
\"max\": 79999.0,\n        \"num_unique_values\": 53,\n
\"samples\": [\n          19499.0,\n          499.0,\n
2999.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"rating\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.20551465276089298,\n        \"min\": 3.7,\n        \"max\":
4.6,\n        \"num_unique_values\": 10,\n        \"samples\": [\n
3.8,\n          4.2,\n          4.4\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"discount\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
35,\n        \"samples\": [\n          \"(75% off)\",\n
\"(17% off)\",\n          \"(62% off)\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"discount_percent\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
15.3680115942171,\n        \"min\": 1.0,\n        \"max\": 87.0,\n
\"num_unique_values\": 34,\n        \"samples\": [\n          24.0,\n
11.0,\n          65.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"price_diff\",\n      \"properties\": {\n        \"dtype\":
```

\"number\",\n   \"std\": 7171.301745403498,\n   \"min\":
9.0,\n   \"max\": 32571.0,\n   \"num_unique_values\": 55,\n
\"samples\": [\n   32571.0,\n   3000.0,\n
1224.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"anomaly_score\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.08199247851754597,\n   \"min\": -
0.14090804134287993,\n   \"max\": 0.2116085948696077,\n
\"num_unique_values\": 80,\n   \"samples\": [\n
0.09884690415474667,\n   0.10273544679423546,\n
0.16395960904250967\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"anomaly_flag\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": -1,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   -1,\n   1\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df_clean"}

```
df_final = df_clean[df_clean['anomaly_flag'] == 1]

df_final.to_csv(
    'smartphoneamazon_clean_no_discrepancies.csv',
    index=False
)

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns



df = pd.read_csv('smartphoneamazon_clean_no_discrepancies.csv')
df.head()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 119,\n \"fields\": [\
n   {\n   \"column\": \"listed_price\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 9780.874930965176,\n
\"min\": 79.0,\n   \"max\": 41999.0,\n
\"num_unique_values\": 49,\n   \"samples\": [\n
24999.0,\n   399.0,\n   901.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"delivery_date\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n   \"samples\": [\n   \"Sat, 20
Dec\",\n   \"20 - 26 Dec\",\n   \"Mon, 22 Dec\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"product_name\",\n
\"properties\": {\n   \"dtype\": \"string\",\n

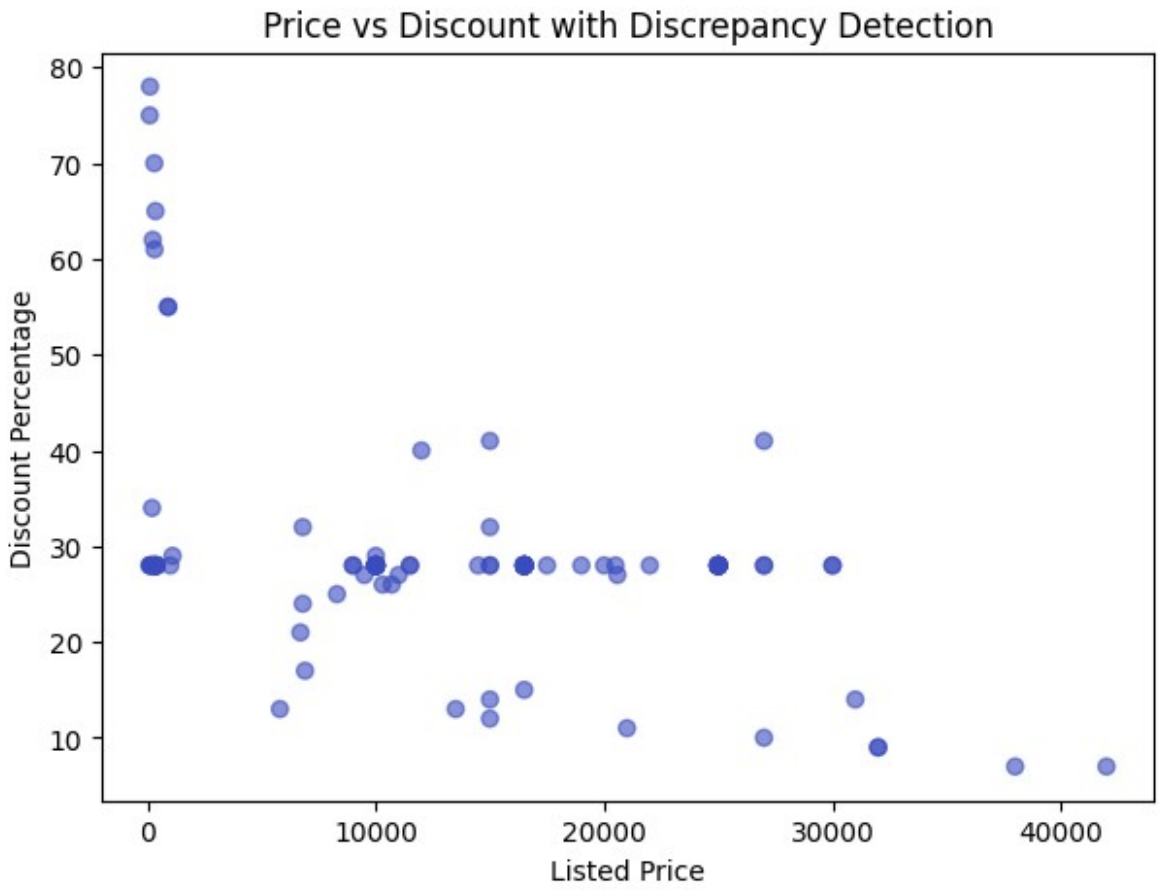
```
\"num_unique_values\": 69,\n        \"samples\": [\n          \"POCO
M6 Plus 5G Ice Silver 8GB RAM 128GB ROM\",\n          \"Samsung Galaxy
A55 5G (Awesome Iceblue, 8GB RAM, 256GB Storage) | AI | Metal Frame |
50 MP Main Camera (OIS) | Super HDR Video| Nightography | IP67 |
Corning Gorilla Glass Victus+ | sAMOLED Display\",\n          \"STRIFF
Smartphone Stand, Tabletop, Foldable, Mobile Phone Stand, Tablet
Stand, Smartphone Holder, Adjustable Height, Anti-Slip, Lightweight,
Compact, Portrait and Horizontal, Easy to Carry(Black)\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"review_count\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2961.8148564659364,\n        \"min\": 1000.0,\n        \"max\":
22700.0,\n        \"num_unique_values\": 37,\n        \"samples\": [\n
5200.0,\n          1000.0,\n          5300.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"mrp\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 14823.29035985531,\n
\"min\": 299.0,\n        \"max\": 48999.0,\n
\"num_unique_values\": 47,\n        \"samples\": [\n
12999.0,\n          995.0,\n          35999.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"rating\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.18952410058626776,\n        \"min\": 3.7,\n        \"max\": 4.6,\n
\"num_unique_values\": 10,\n        \"samples\": [\n          3.8,\n
4.2,\n          4.4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"discount\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 27,\n
\"samples\": [\n          \"(28% off)\",\n          \"(24% off)\",\n
\"(7% off)\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"discount_percent\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 11.524464172460563,\n        \"min\":
7.0,\n        \"max\": 78.0,\n        \"num_unique_values\": 27,\n
\"samples\": [\n          29.0,\n          24.0,\n          7.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"price_diff\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
6191.038766783945,\n        \"min\": 50.0,\n        \"max\": 19000.0,\
n        \"num_unique_values\": 44,\n        \"samples\": [\n
320.0,\n          520.0,\n          400.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"anomaly_score\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.055499470471804736,\n        \"min\": 0.0128753327570785,\n
\"max\": 0.2116085948696077,\n        \"num_unique_values\": 67,\n
\"samples\": [\n          0.0128753327570785,\n
0.0406748456894724,\n          0.0695767206410586\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"anomaly_flag\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 1,\n   \"max\": 1,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"df"}

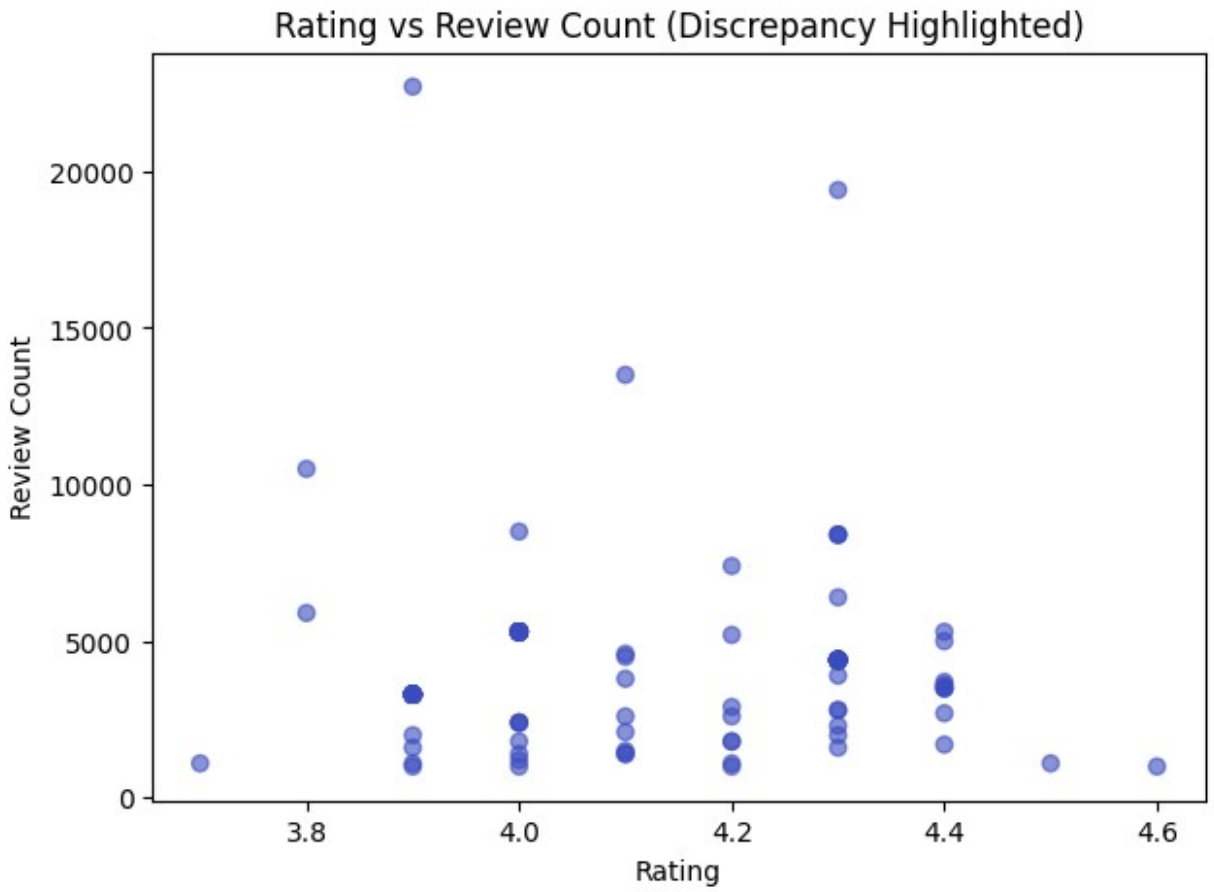
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119 entries, 0 to 118
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   listed_price      119 non-null    float64
 1   delivery_date     118 non-null    object 
 2   product_name      119 non-null    object 
 3   review_count      119 non-null    float64
 4   mrp               119 non-null    float64
 5   rating            119 non-null    float64
 6   discount          39 non-null     object 
 7   discount_percent  119 non-null    float64
 8   price_diff        119 non-null    float64
 9   anomaly_score     119 non-null    float64
 10  anomaly_flag      119 non-null    int64  
dtypes: float64(7), int64(1), object(3)
memory usage: 10.4+ KB
```

```
plt.figure(figsize=(7,5))
plt.scatter(
    df['listed_price'],
    df['discount_percent'],
    c=df['anomaly_flag'],
    cmap='coolwarm',
    alpha=0.6
)
plt.xlabel('Listed Price')
plt.ylabel('Discount Percentage')
plt.title('Price vs Discount with Discrepancy Detection')
plt.show()
```

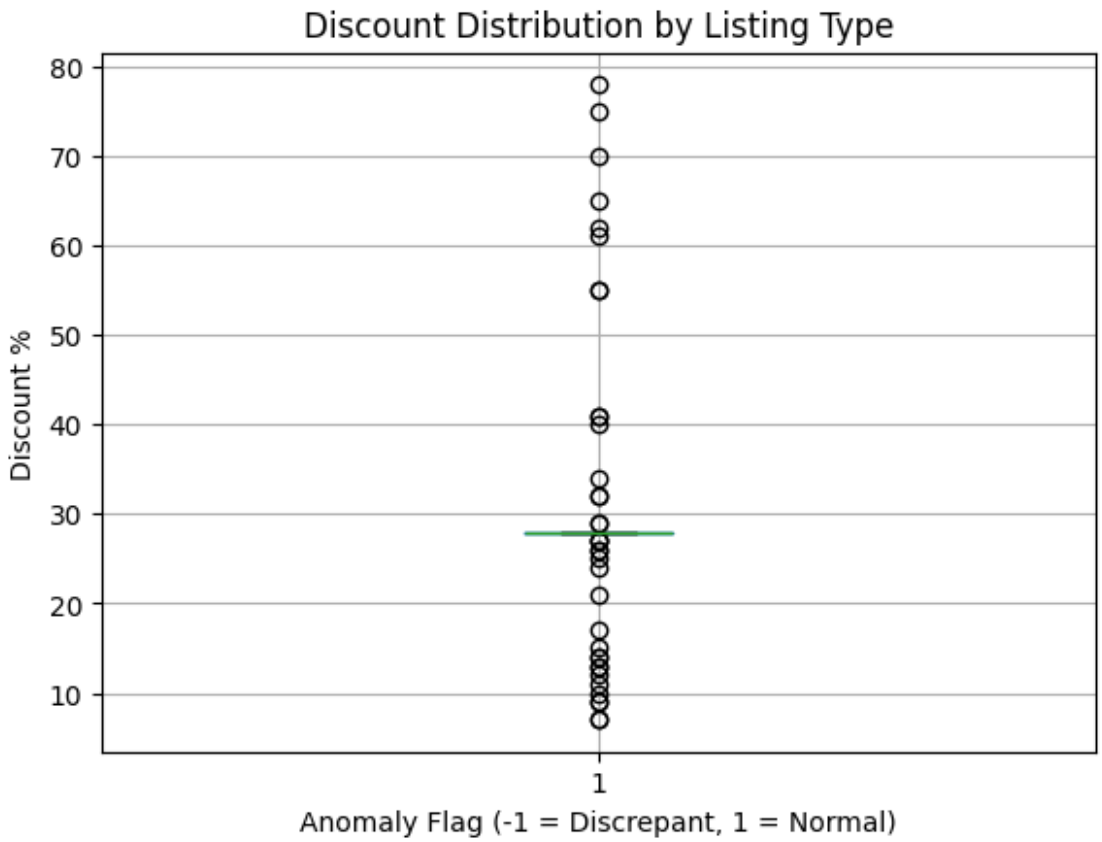



```
plt.figure(figsize=(7,5))
plt.scatter(
    df['rating'],
    df['review_count'],
    c=df['anomaly_flag'],
    cmap='coolwarm',
    alpha=0.6
)
plt.xlabel('Rating')
plt.ylabel('Review Count')
plt.title('Rating vs Review Count (Discrepancy Highlighted)')
plt.show()
```

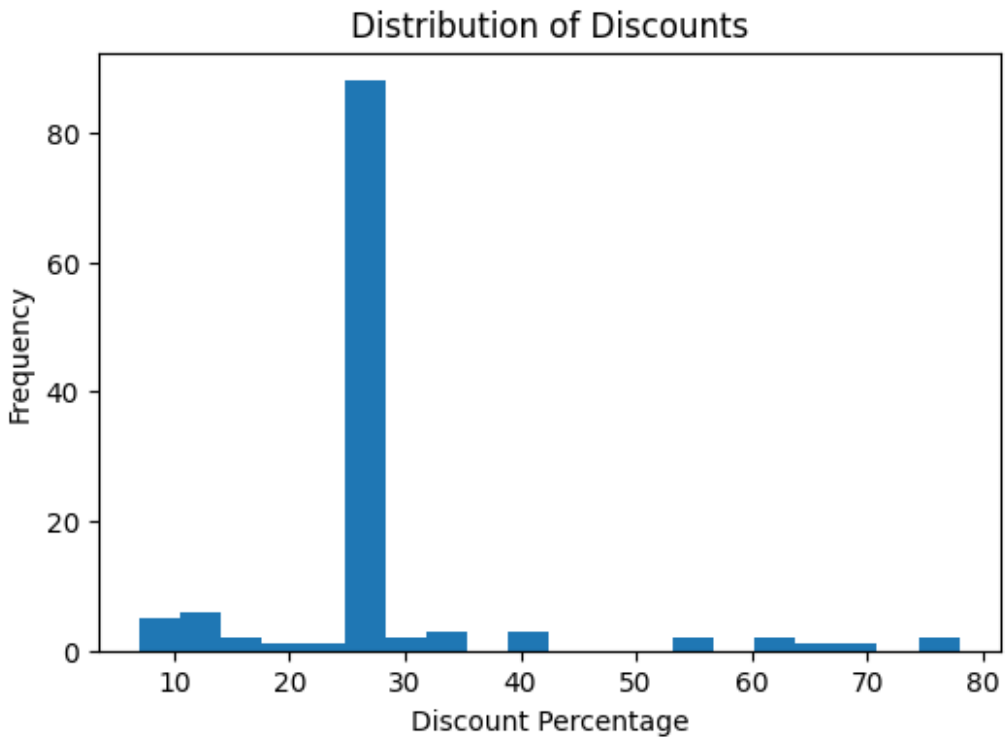



```
plt.figure(figsize=(6,4))
df.boxplot(column='discount_percent', by='anomaly_flag')
plt.title('Discount Distribution by Listing Type')
plt.suptitle('')
plt.xlabel('Anomaly Flag (-1 = Discrepant, 1 = Normal)')
plt.ylabel('Discount %')
plt.show()
```

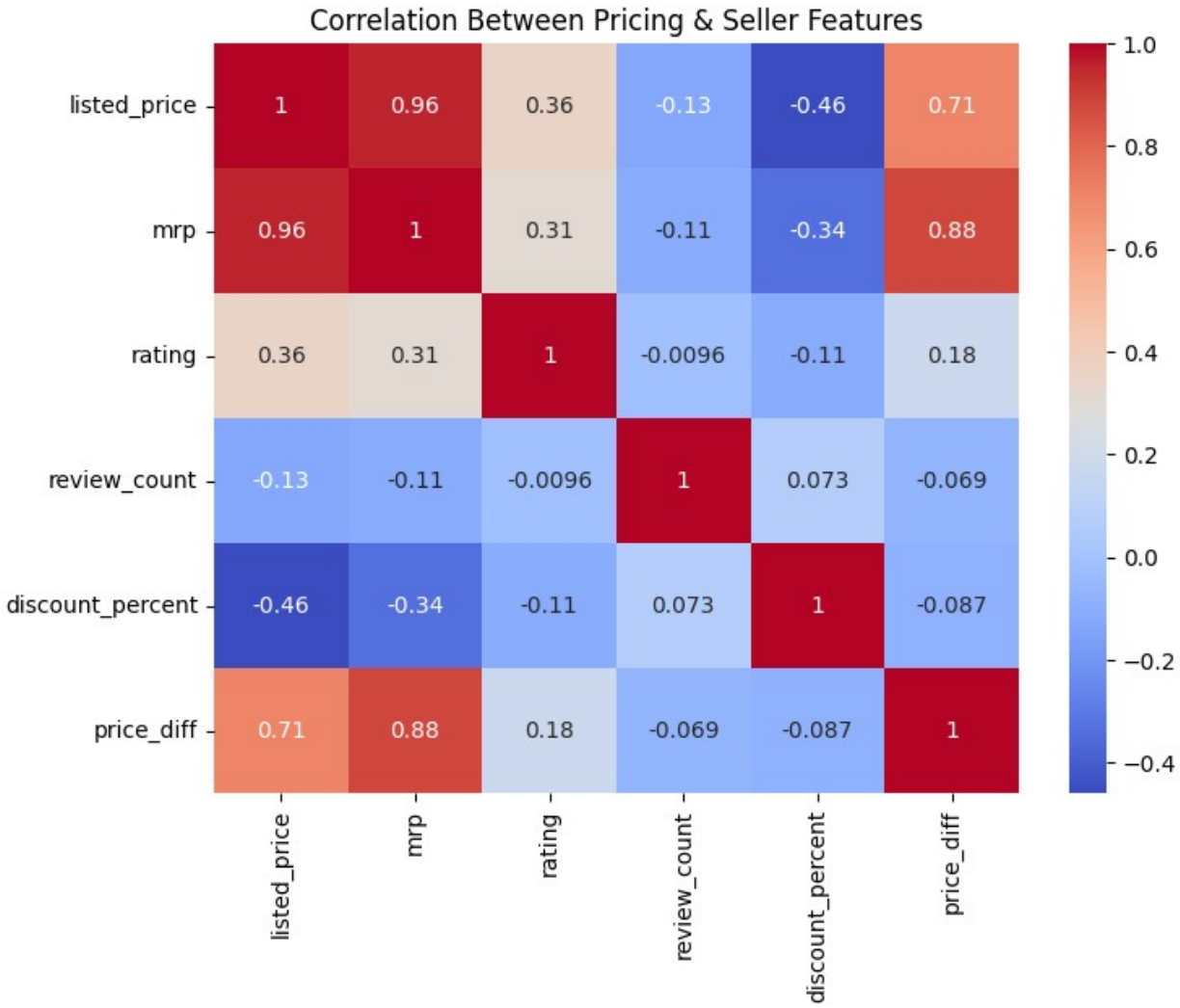
```
<Figure size 600x400 with 0 Axes>
```




```
plt.figure(figsize=(6,4))
plt.hist(df['discount_percent'], bins=20)
plt.xlabel('Discount Percentage')
plt.ylabel('Frequency')
plt.title('Distribution of Discounts')
plt.show()
```




```
plt.figure(figsize=(8,6))
sns.heatmap(
    df[['listed_price','mrp','rating',
        'review_count','discount_percent','price_diff']].corr(),
    annot=True,
    cmap='coolwarm'
)
plt.title('Correlation Between Pricing & Seller Features')
plt.show()
```


Correlation Between Pricing & Seller Features
listed_price
mrp
rating
review_count
discount_percent
price_diff
1
0.96
0.36
-0.13
-0.46
0.71
0.96
1
0.31
-0.11
-0.34
0.88
0.36
0.31
1
-0.0096
-0.11
0.18
-0.13
-0.11
-0.0096
1
0.073
-0.069
-0.46
-0.34
-0.11
0.073
1
-0.087
0.71
0.88
0.18
-0.069
-0.087
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4