

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style("darkgrid")
plt.rcParams['figure.figsize'] = (12, 6)

df = pd.read_csv('/content/drive/MyDrive/datat/amazon.csv') # Corrected file path
```

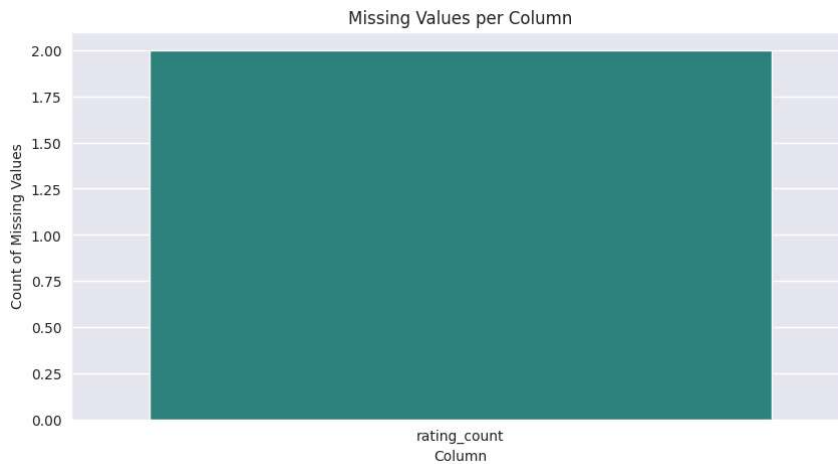
```
missing_values = df.isnull().sum()
missing_values = missing_values[missing_values > 0].sort_values(ascending=False)

plt.figure(figsize=(10, 5))
sns.barplot(x=missing_values.index, y=missing_values.values, palette="viridis")
plt.title('Missing Values per Column')
plt.xlabel('Column')
plt.ylabel('Count of Missing Values')
plt.show()
#
```

/tmp/ipython-input-2646896200.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(x=missing_values.index, y=missing_values.values, palette="viridis")
```



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style("darkgrid")
plt.rcParams['figure.figsize'] = (12, 6)
```

```
import pandas as pd
import numpy as np

def clean_and_convert_price(price_series):
    return pd.to_numeric(price_series.astype(str).str.replace('$', '', regex=False).str.replace(',', '', regex=False), errors='coerce')

def clean_and_convert_percentage(percentage_series):
    return pd.to_numeric(percentage_series.astype(str).str.replace('%', '', regex=False), errors='coerce')

def clean_and_convert_rating(rating_series):
    rating_series = rating_series.astype(str).apply(lambda x: x.split(' ')[0] if ' out of ' in x else x)
    return pd.to_numeric(rating_series, errors='coerce')

def clean_and_convert_rating_count(count_series):
    return pd.to_numeric(count_series.astype(str).str.replace(', ', '', regex=False), errors='coerce')

df['discounted_price'] = clean_and_convert_price(df['discounted_price'])
df['actual_price'] = clean_and_convert_price(df['actual_price'])
df['discount_percentage'] = clean_and_convert_percentage(df['discount_percentage'])
df['rating'] = clean_and_convert_rating(df['rating'])
df['rating_count'] = clean_and_convert_rating_count(df['rating_count'])
numerical_cols = ['discounted_price', 'actual_price', 'discount_percentage', 'rating', 'rating_count']
categorical_cols = ['category']

for col in numerical_cols:
    if col in df.columns and df[col].isnull().any():
        median_value = df[col].median()
        df.loc[:, col] = df[col].fillna(median_value)

for col in categorical_cols:
    if col in df.columns and df[col].isnull().any():
        mode_value = df[col].mode()
        # Use .loc to avoid FutureWarning with inplace=True and ensuring mode_value is not empty
        df.loc[:, col] = df[col].fillna(mode_value[0] if not mode_value.empty else 'Unknown')

print("Missing value imputation completed after cleaning.")
print(df.isnull().sum())
```

Missing value imputation completed after cleaning.

```
product_id      0
product_name    0
category        0
discounted_price 0
actual_price    0
discount_percentage 0
rating          0
rating_count    0
about_product   0
user_id         0
user_name       0
review_id       0
review_title    0
review_content  0
img_link        0
product_link    0
```

dtype: int64

```
plt.figure(figsize=(8, 6))
sns.boxplot(y=df['discounted_price'])
plt.title('Box Plot for Discounted Price (Before Treatment)')
plt.ylabel('discounted_price')
plt.show()
```



```
col_to_clean = 'discounted_price'
df['price_cleaned'] = df[col_to_clean]

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

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

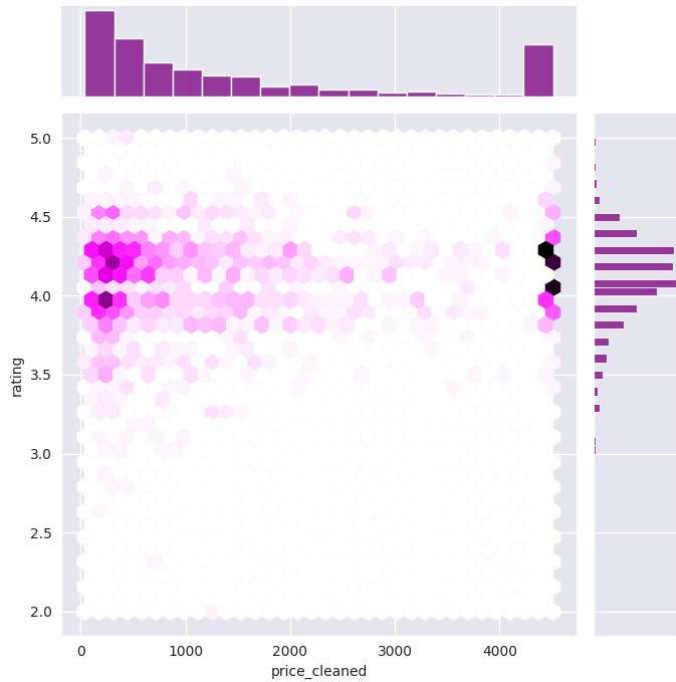
df['price_cleaned'] = np.where(df['price_cleaned'] > upper_bound, upper_bound, df['price_cleaned'])
df['price_cleaned'] = np.where(df['price_cleaned'] < lower_bound, lower_bound, df['price_cleaned'])

plt.figure(figsize=(8, 6))
sns.boxplot(y=df['price_cleaned'])
plt.title('Box Plot for Discounted Price (After Treatment)')
plt.ylabel('price_cleaned')
plt.show()
#
```



```
sns.jointplot(x='price_cleaned', y='rating', data=df, kind='hex', height=7, color='purple')
plt.suptitle('Joint Distribution: Cleaned Price vs. Rating', y=1.02)
plt.show()
#
```

Joint Distribution: Cleaned Price vs. Rating



```
# Select the top 5 categories for a clearer visualization
top_categories = df['category'].value_counts().nlargest(5).index
df_top_cat = df[df['category'].isin(top_categories)]

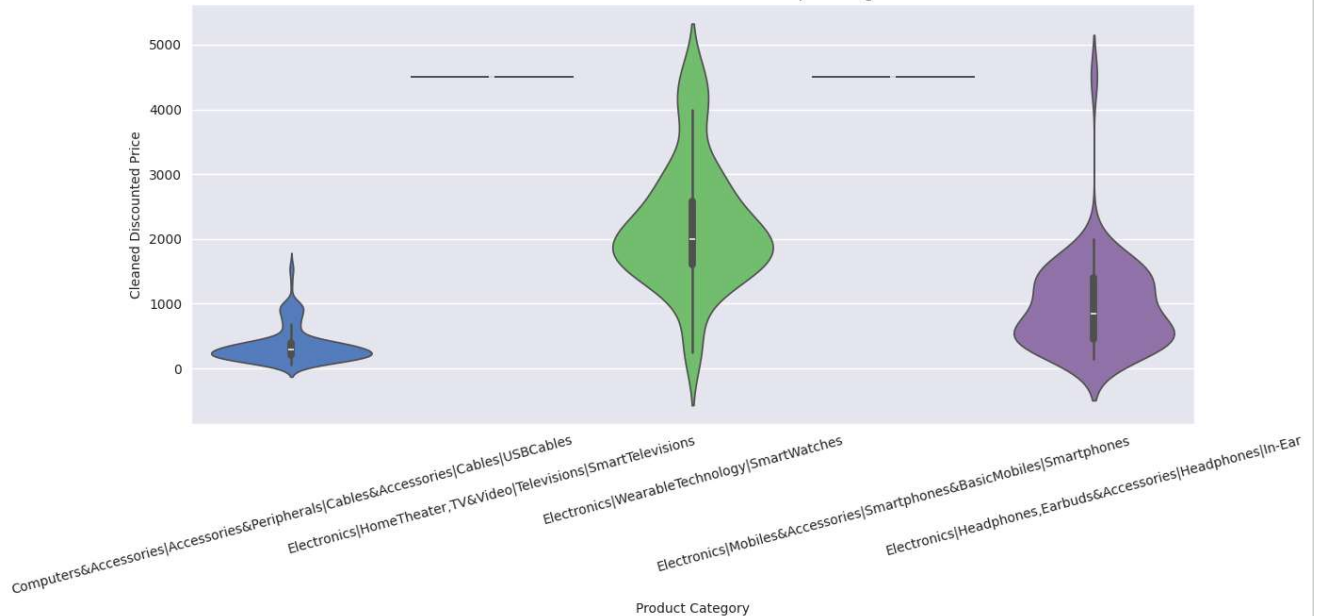
plt.figure(figsize=(14, 7))
sns.violinplot(x='category', y='price_cleaned', data=df_top_cat, palette='muted')
plt.title('Distribution of Discounted Price Across Top 5 Categories')
plt.xlabel('Product Category')
plt.ylabel('Cleaned Discounted Price')
plt.xticks(rotation=15)
plt.tight_layout()
plt.show()
#
```

/tmp/ipython-input-1282630306.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.violinplot(x='category', y='price_cleaned', data=df_top_cat, palette='muted')
```

Distribution of Discounted Price Across Top 5 Categories



```
# 1. Price Difference: (السعر الأصلي - السعر المخفض)
df['price_difference'] = df['actual_price'] - df['discounted_price']
```

```
# 2. Log Transformation: (التحويل اللوغاريتمي للسعر المخفض للتظليل)
# لتجنب أخطاء اللوغاريتم في حالة وجود قيم صفرية نستخدم np.log1p
df['log_price_cleaned'] = np.log1p(df['price_cleaned'])
```

```
# 3. Review Density: (كثافة المراجعات / التقييم)
# سنفترض تقييماً 1 إذا كان مفقوداً أو صفراً) NaN لتجنب القسمة على صفر أو
df['rating_safe'] = df['rating'].apply(lambda x: x if x > 0 else 1)
df['review_density'] = df['rating_count'] / df['rating_safe']
```

```
# تحديد أهم 4 فئات
top_categories = df['category'].value_counts().nlargest(4).index
```

```

# تطبيق التشفير (One-Hot Encoding)
df_encoded = pd.get_dummies(df, columns=['category'], prefix='is', dummy_na=False)

# بناء قائمة الأعمدة المشفرة
category_cols = [f'is_{cat}' for cat in top_categories]

# تصفية الأعمدة للتأكد من وجودها بعد التشفير
new_encoded_cols = [col for col in category_cols if col in df_encoded.columns]

# الأعمدة الأصلية التي كانت موجودة وتم تنظيفها
original_cols = ['price_cleaned', 'actual_price', 'discount_percentage', 'rating', 'rating_count']

# الأعمدة الجديدة المشتقة
engineered_cols = ['price_difference', 'log_price_cleaned', 'review_density']

# القائمة النهائية لجميع الأعمدة في مصفوفة الارتباط
final_correlation_cols = original_cols + engineered_cols + new_encoded_cols

# بناء مصفوفة الارتباط الموسعة ورسمها
correlation_matrix = df_encoded[final_correlation_cols].corr()

plt.figure(figsize=(14, 12))
sns.heatmap(correlation_matrix,
            annot=True,
            cmap='viridis', # تغيير نمط الألوان
            fmt=".2f",
            linewidths=.5,
            cbar_kws={'label': 'Correlation Coefficient'})

plt.title('Expanded Correlation Matrix (11x11) including Engineered Features')
plt.show()
#

```

