dacanay-sa1-dsc1105

May 14, 2025

1 I. Data Preview

1.1 A. First Five Rows

[1]:
```python
import pandas as pd

df = pd.read_csv("/kaggle/input/ecommerce-assessment/EDA_Ecommerce_Assessment.
 ↪csv")
df.head()
```

[1]:
```
   Customer_ID  Gender  Age  Browsing_Time  Purchase_Amount  Number_of_Items  \
0            1    Male   65          46.55           231.81                6
1            2  Female   19          98.80           472.78                8
2            3    Male   23          79.48           338.44                1
3            4    Male   45          95.75            37.13                7
4            5    Male   46          33.36           235.53                3

   Discount_Applied  Total_Transactions        Category  Satisfaction_Score
0                17                  16        Clothing                   2
1                15                  43           Books                   4
2                28                  31     Electronics                   1
3                43                  27  Home & Kitchen                   5
4                10                  33           Books                   3
```

1.2 B. Number of Rows and Columns

[2]:
```python
df.shape
```

[2]:
```
(3000, 10)
```

1.3 C. Columns, Non-Null Values, and Data Types Information

[3]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
```

```
---  ------              --------------  -----
 0   Customer_ID         3000 non-null   int64
 1   Gender              3000 non-null   object
 2   Age                 3000 non-null   int64
 3   Browsing_Time       3000 non-null   float64
 4   Purchase_Amount     3000 non-null   float64
 5   Number_of_Items     3000 non-null   int64
 6   Discount_Applied    3000 non-null   int64
 7   Total_Transactions  3000 non-null   int64
 8   Category            3000 non-null   object
 9   Satisfaction_Score  3000 non-null   int64
dtypes: float64(2), int64(6), object(2)
memory usage: 234.5+ KB
```

1.4 D. Summary per Column

[4]:
```
df.describe(include='all')
```

```
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1458:
RuntimeWarning: invalid value encountered in greater
  has_large_values = (abs_vals > 1e6).any()
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1459:
RuntimeWarning: invalid value encountered in less
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any()
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1459:
RuntimeWarning: invalid value encountered in greater
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any()
```

[4]:
```
        Customer_ID Gender          Age  Browsing_Time  Purchase_Amount  \
count   3000.000000   3000  3000.000000    3000.000000      3000.000000
unique          NaN      2          NaN            NaN              NaN
top             NaN   Male          NaN            NaN              NaN
freq            NaN   1517          NaN            NaN              NaN
mean    1500.500000    NaN    43.606000      59.868937       247.962540
std      866.169729    NaN    14.963759      34.293489       140.875783
min        1.000000    NaN    18.000000       1.000000         5.030000
25%      750.750000    NaN    31.000000      29.985000       128.695000
50%     1500.500000    NaN    44.000000      59.160000       245.090000
75%     2250.250000    NaN    57.000000      89.330000       367.200000
max     3000.000000    NaN    69.000000     119.950000       499.610000

        Number_of_Items  Discount_Applied  Total_Transactions  Category  \
count       3000.000000       3000.000000         3000.000000      3000
unique              NaN               NaN                 NaN         5
top                 NaN               NaN                 NaN  Clothing
freq                NaN               NaN                 NaN       629
mean           4.989667         24.345000           24.683000       NaN
```

```
std            2.561200     14.709433      14.214518     NaN
min            1.000000      0.000000       1.000000     NaN
25%            3.000000     12.000000      12.000000     NaN
50%            5.000000     24.000000      24.000000     NaN
75%            7.000000     37.000000      37.000000     NaN
max            9.000000     49.000000      49.000000     NaN

        Satisfaction_Score
count          3000.000000
unique                 NaN
top                    NaN
freq                   NaN
mean              3.066000
std               1.402723
min               1.000000
25%               2.000000
50%               3.000000
75%               4.000000
max               5.000000
```

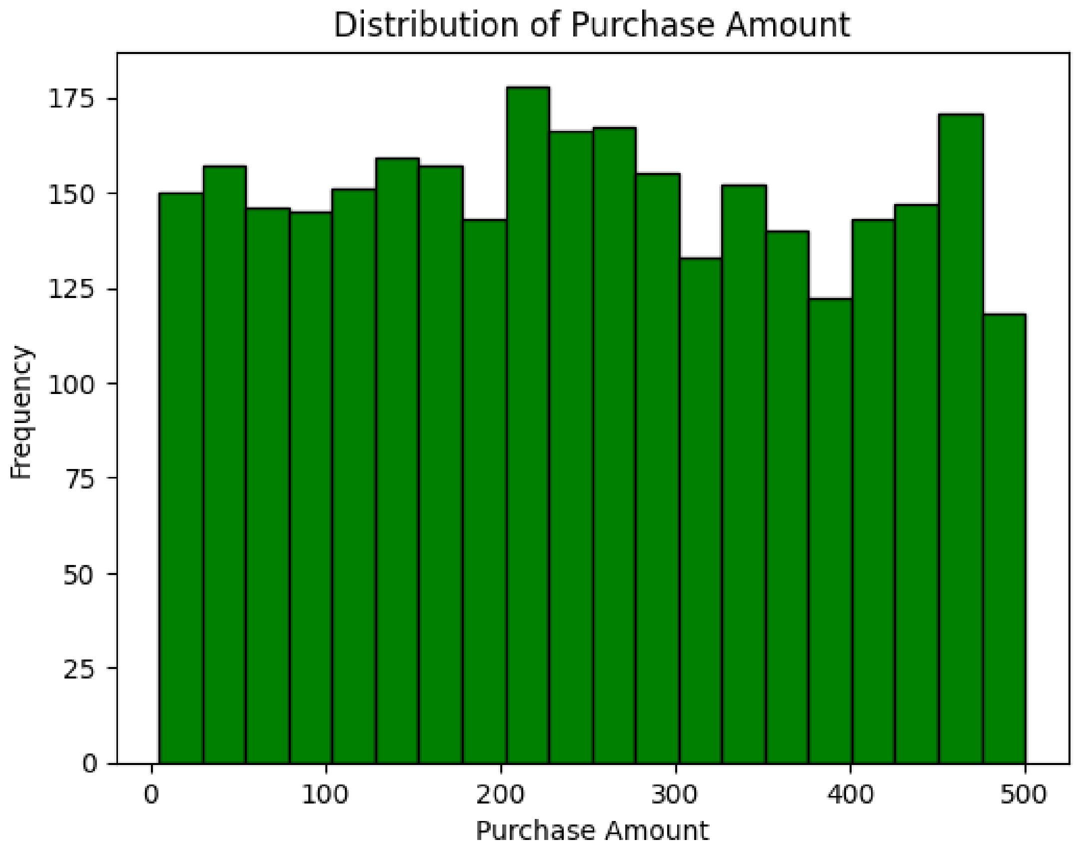
2 II. Univariate Data Analysis

2.1 A. Data Visualization

2.1.1 1. Purchase Amount (Total Amount Spent)

[5]:

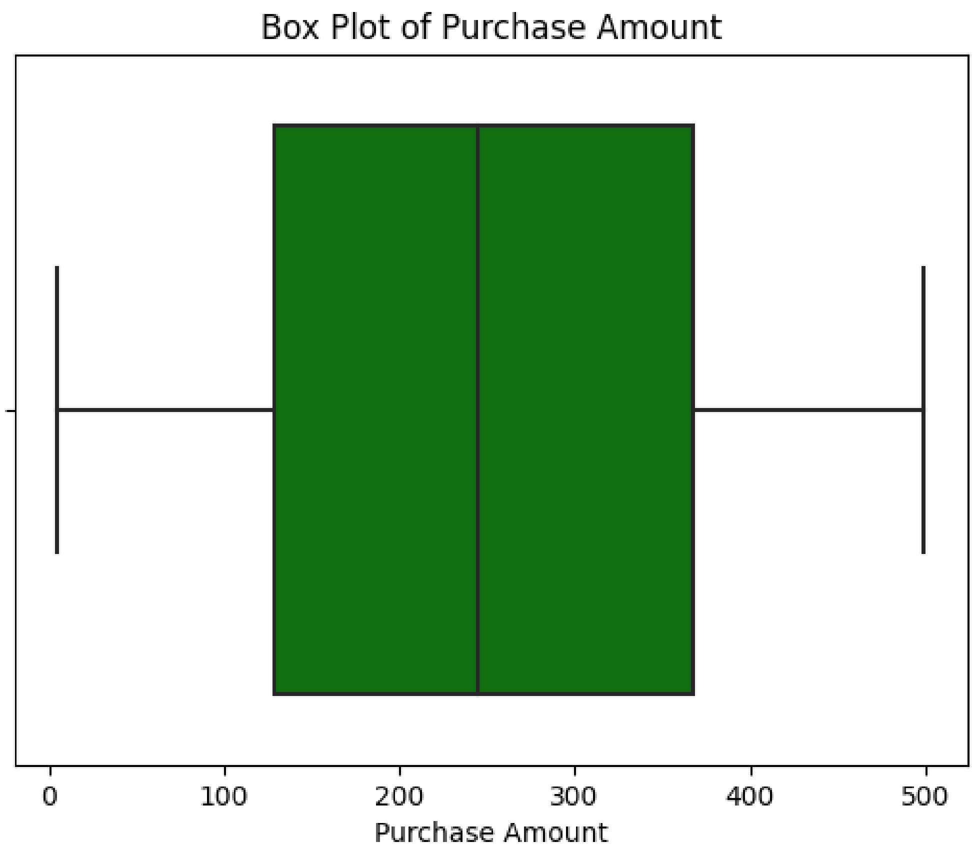
```
import matplotlib.pyplot as plt
from IPython.display import display, Markdown

# Histogram
plt.hist(df['Purchase_Amount'], bins=20, color='green', edgecolor='black')
plt.title('Distribution of Purchase Amount')
plt.xlabel('Purchase Amount')
plt.ylabel('Frequency')
plt.show()

display(Markdown('''
The figure above illustrates a histogram showing the distribution of purchase␣
↪amounts ranging from \\$5 to \\\$500.
As shown, most customers have purchases between \\\$200 and \\\$225, while the␣
↪fewest fall within
the \\\$475 to \\\$500 range.'''))
```



The figure above illustrates a histogram showing the distribution of purchase amounts ranging from \$5 to \$500. As shown, most customers have purchases between \$200 and \$225, while the fewest fall within the \$475 to \$500 range.

[6]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Single color box
sns.boxplot(x=df['Purchase_Amount'], color='green')
plt.title('Box Plot of Purchase Amount')
plt.xlabel('Purchase Amount')
plt.show()

display(Markdown('''
The figure above illustrates a box plot showing the distribution of purchase␣
↪amounts. As shown,
the distribution of purchase amounts appears to be approximately normal, as␣
↪indicated by the symmetrical shape
of the box plot. The 25th percentile is approximately \\$120, indicating that a␣
↪quarter of the customers
```

```
have purchases below this value. The 50th percentile, or median, is nearly␣
  ↪\\\$250, meaning that half
of the customers fall below this amount. The 75th percentile is approximately␣
  ↪\\\$380, suggesting that 75% of the customers
have purchases under this value. Lastly, the graph shows that there are no␣
  ↪outliers in the distribution.'''))
```



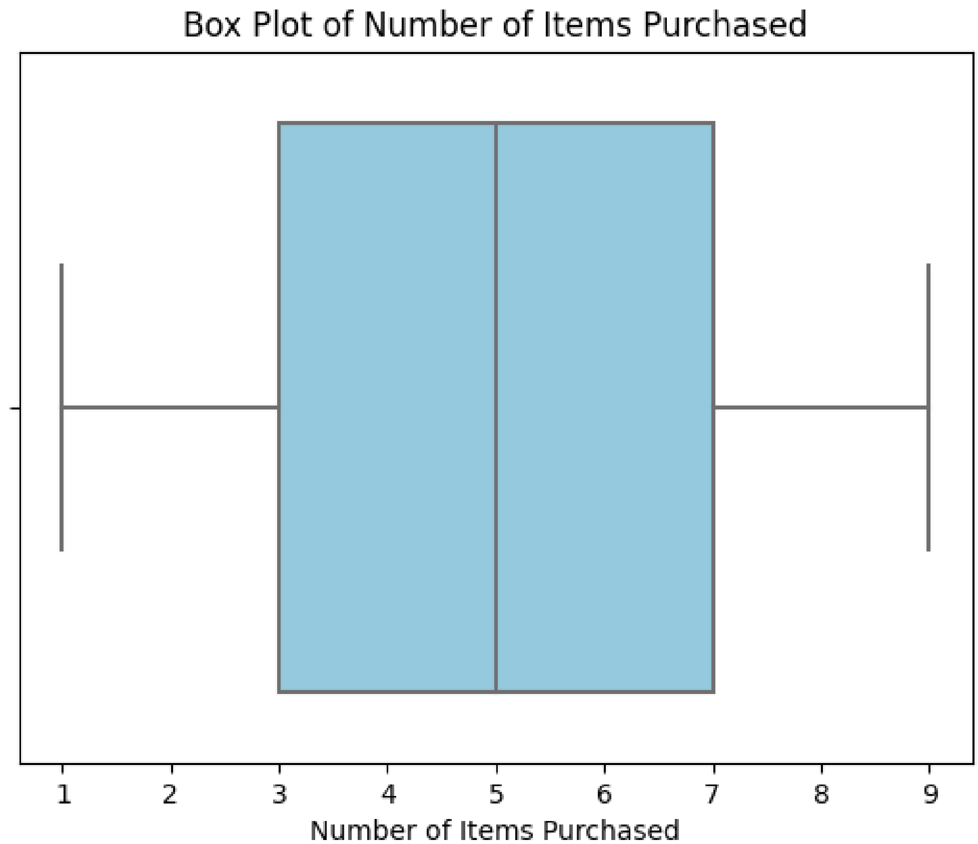
The figure above illustrates a box plot showing the distribution of purchase amounts. As shown, the distribution of purchase amounts appears to be approximately normal, as indicated by the symmetrical shape of the box plot. The 25th percentile is approximately \$120, indicating that a quarter of the customers have purchases below this value. The 50th percentile, or median, is nearly \\$250, meaning that half of the customers fall below this amount. The 75th percentile is approximately \\$380, suggesting that 75% of the customers have purchases under this value. Lastly, the graph shows that there are no outliers in the distribution.

2.1.2 2. Number of Items Purchased

[7]:
```
import matplotlib.pyplot as plt
import numpy as np

# Histogram
plt.hist(df['Number_of_Items'], bins=9, color='skyblue', edgecolor='black')
plt.title('Distribution of Number of Items Purchased')
plt.xlabel('Number of Items Purchased')
plt.ylabel('Frequency')
plt.show()

display(Markdown('''
The figure above illustrates a histogram showing the distribution of the number␣
↪of items purchased
per transaction, ranging from one to nine items. As shown, most customers␣
↪purchased four items in a single transaction,
while the fewest purchased only three.'''))
```

The figure above illustrates a histogram showing the distribution of the number of items purchased per transaction, ranging from one to nine items. As shown, most customers purchased four items in a single transaction, while the fewest purchased only three.

[8]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Single color box
sns.boxplot(x=df['Number_of_Items'], color='skyblue')
plt.title('Box Plot of Number of Items Purchased')
plt.xlabel('Number of Items Purchased')
plt.show()

display(Markdown('''
The figure above illustrates a box plot showing the distribution of the number␣
↪of items purchased per transaction.
As shown, the distribution appears to be approximately normal, as indicated by␣
↪the symmetrical shape of the box plot.
The 25th percentile is 3, meaning that 25% of customers purchased three items␣
↪or fewer in a single transaction.
The median (50th percentile) is 5, indicating that half of the customers␣
↪purchased five items or fewer.
The 75th percentile is 7, suggesting that 75% of customers purchased seven␣
↪items or fewer.
Lastly, the absence of outliers in the graph indicates a relatively consistent␣
↪pattern of purchases.'''))
```

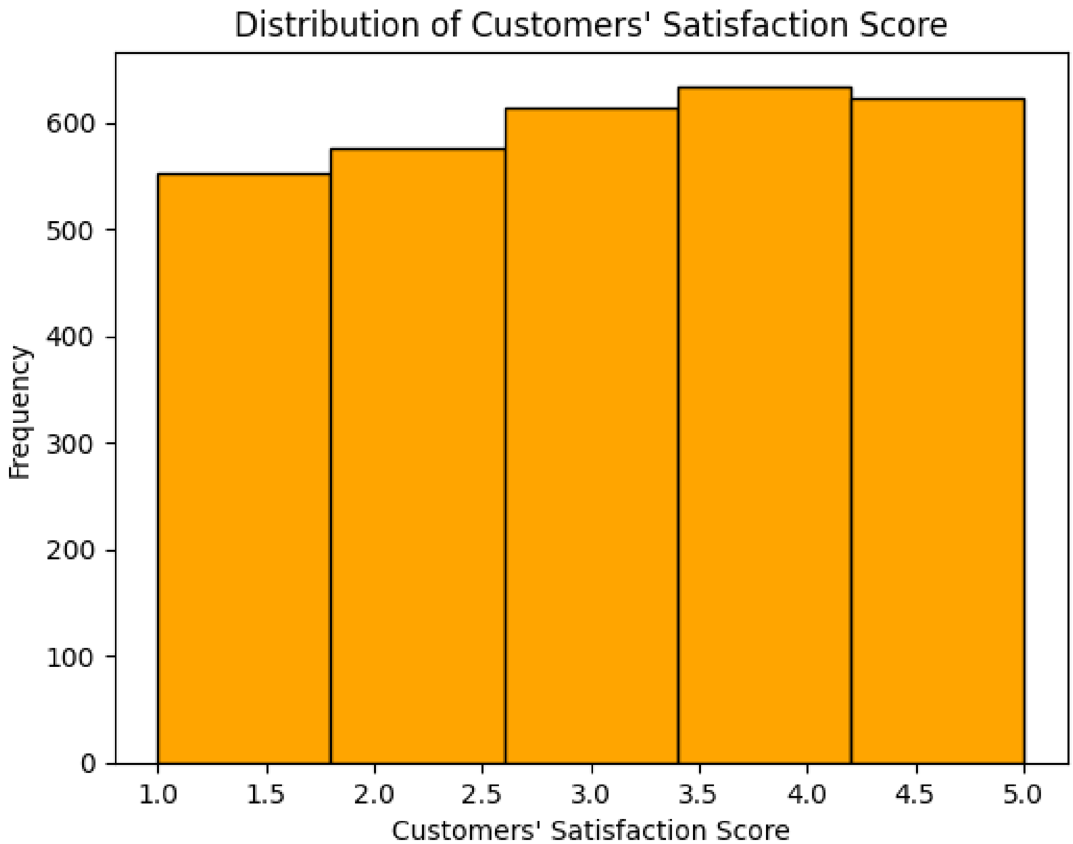


The figure above illustrates a box plot showing the distribution of the number of items purchased per transaction. As shown, the distribution appears to be approximately normal, as indicated by the symmetrical shape of the box plot. The 25th percentile is 3, meaning that 25% of customers purchased three items or fewer in a single transaction. The median (50th percentile) is 5, indicating that half of the customers purchased five items or fewer. The 75th percentile is 7, suggesting that 75% of customers purchased seven items or fewer. Lastly, the absence of outliers in the graph indicates a relatively consistent pattern of purchases.

2.1.3 3. Customers' Satisfaction Score

[9]:
```
import matplotlib.pyplot as plt
import numpy as np

# Histogram
plt.hist(df['Satisfaction_Score'], bins=5, color='orange', edgecolor='black')
plt.title("Distribution of Customers' Satisfaction Score")
plt.xlabel("Customers' Satisfaction Score")
plt.ylabel('Frequency')
plt.show()
```

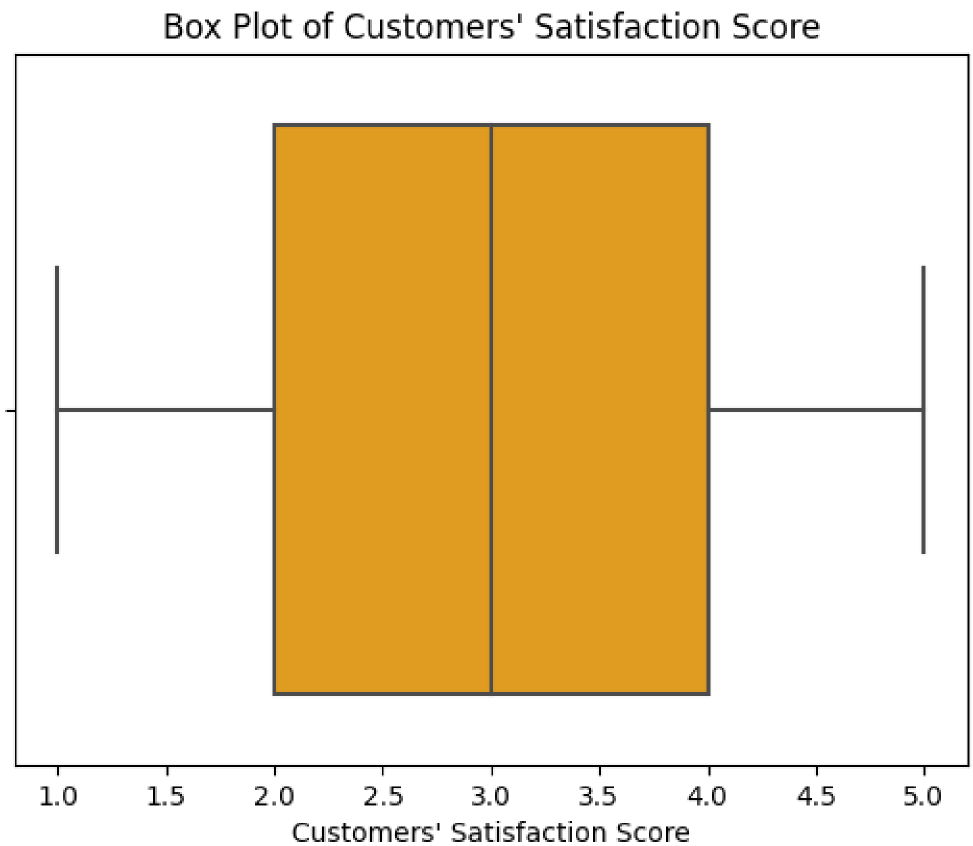
```
display(Markdown('''
The figure above illustrates a histogram showing the distribution of customer␣
↪satisfaction scores,
ranging from a scale of one to five. As shown, the majority of customers rated␣
↪their satisfaction between three and five,
indicating a generally high level of satisfaction. In contrast, a smaller but␣
↪close number of customers gave ratings of one or two.'''))
```



The figure above illustrates a histogram showing the distribution of customer satisfaction scores, ranging from a scale of one to five. As shown, the majority of customers rated their satisfaction between three and five, indicating a generally high level of satisfaction. In contrast, a smaller but close number of customers gave ratings of one or two.

[10]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Single color box
sns.boxplot(x=df['Satisfaction_Score'], color='orange')
plt.title("Box Plot of Customers' Satisfaction Score")
plt.xlabel("Customers' Satisfaction Score")
```

```
plt.show()

display(Markdown('''
The figure above illustrates a box plot showing the distribution of total␣
  ↪purchase amounts. As shown,
the distribution of purchase amounts appears to be approximately normal, as␣
  ↪indicated by the symmetrical shape
of the box plot. The 25th percentile is approximately \\$120, indicating that a␣
  ↪quarter of the customers
have total purchases below this value. The 50th percentile, or median, is␣
  ↪nearly \\\$250, meaning that half
of the customers fall below this amount. The 75th percentile is approximately␣
  ↪\\\$380, suggesting that 75% of the customers
have total purchases under this value. Lastly, the graph shows that there are␣
  ↪no outliers in the distribution.'''))
```



The figure above illustrates a box plot showing the distribution of total purchase amounts. As shown, the distribution of purchase amounts appears to be approximately normal, as indicated by the symmetrical shape of the box plot. The 25th percentile is approximately \$120, indicating that a quarter of the customers have total purchases below this value. The 50th percentile, or median, is nearly \$250, meaning that half of the customers fall below this amount. The 75th percentile is approximately \$380, suggesting that 75% of the customers have total purchases under this value. Lastly, the graph shows that there are no outliers in the distribution.

2.2 B. Central Tendency and Spread of the Customers' Purchase Amount

[11]:
```
purchase_amount = df['Purchase_Amount']

# Central Tendency
central_tendency = {
    'Mean': purchase_amount.mean(),
    'Median': purchase_amount.median(),
    'Mode': purchase_amount.mode().iloc[0]  # first mode if multiple
}

# Spread
spread = {
    'Variance': purchase_amount.var(),
    'Standard Deviation': purchase_amount.std(),
    'IQR': purchase_amount.quantile(0.75) - purchase_amount.quantile(0.25)
}

# Result
summary_stats = {**central_tendency, **spread}
summary_df = pd.DataFrame(list(summary_stats.items()), columns=['Statistic',␣
↪'Value'])

# Summary table with an asterisk for 'Mode'
summary_df_mode_star = summary_df.copy()
summary_df_mode_star.loc[summary_df_mode_star['Statistic'] == 'Mode',␣
↪'Statistic'] = 'Mode*'

display(summary_df_mode_star)
display(Markdown("**There are multiple modes in the data.*"))
display(Markdown('''
#### Interpretations
The **mean** purchase amount is approximately 247.96, while the **median** is␣
↪slightly lower at 245.09,
suggesting a fairly symmetrical distribution with a slight right skew.
However, the **mode** is significantly lower at 29.33, and the asterisk␣
↪indicates the presence of multiple modes-implying
that there are distinct clusters of common purchase amounts, likely one around␣
↪very low spending and another in the mid-range.

The **variance** (19,846) and **standard deviation** (140.88) indicate a high␣
↪level of spread in the data,
```

```
suggesting that customer purchase behavior is diverse, with amounts ranging␣
↪widely around the mean.
The **interquartile range (IQR)** of 238.51 confirms this spread, showing that␣
↪the middle 50% of purchase amounts span a wide range.
'''))
```

```
            Statistic         Value
0                Mean    247.962540
1              Median    245.090000
2               Mode*     29.330000
3            Variance  19845.986209
4  Standard Deviation    140.875783
5                 IQR    238.505000
```

**There are multiple modes in the data.*

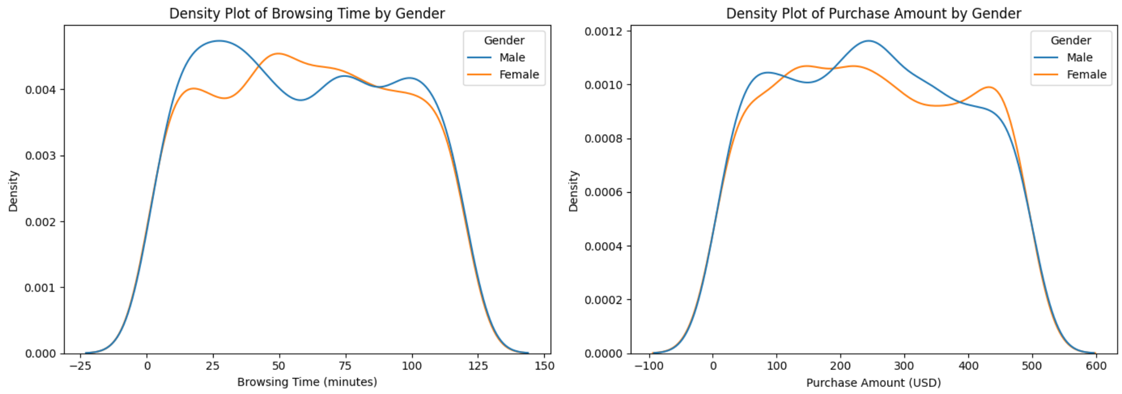
Interpretations The **mean** purchase amount is approximately 247.96, while the **median** is slightly lower at 245.09, suggesting a fairly symmetrical distribution with a slight right skew. However, the **mode** is significantly lower at 29.33, and the asterisk indicates the presence of multiple modes—implying that there are distinct clusters of common purchase amounts, likely one around very low spending and another in the mid-range.

The **variance** (19,846) and **standard deviation** (140.88) indicate a high level of spread in the data, suggesting that customer purchase behavior is diverse, with amounts ranging widely around the mean. The **interquartile range (IQR)** of 238.51 confirms this spread, showing that the middle 50% of purchase amounts span a wide range.

2.3 C. Comparison of Browsing Time and Total Purchase Amount Distribution

[12]:
```
import numpy as np
import warnings

# Data Cleaning and Filtering
df['Browsing_Time'] = pd.to_numeric(df['Browsing_Time'], errors='coerce')
df['Purchase_Amount'] = pd.to_numeric(df['Purchase_Amount'], errors='coerce')
df_cleaned = df[df['Gender'].isin(['Male', 'Female'])].
↪dropna(subset=['Browsing_Time', 'Purchase_Amount'])

# Ensuring Correct Data Types
df_cleaned['Browsing_Time'] = df_cleaned['Browsing_Time'].astype(float)
df_cleaned['Purchase_Amount'] = df_cleaned['Purchase_Amount'].astype(float)

# Suppress warnings
with warnings.catch_warnings():
    warnings.simplefilter(action='ignore', category=FutureWarning)

    # Your plotting code here
    fig, axes = plt.subplots(1, 2, figsize=(14, 5))
```

```python
    sns.kdeplot(data=df_cleaned, x='Browsing_Time', hue='Gender', ax=axes[0])
    axes[0].set_title('Density Plot of Browsing Time by Gender')
    axes[0].set_xlabel('Browsing Time (minutes)')
    axes[0].set_ylabel('Density')

    sns.kdeplot(data=df_cleaned, x='Purchase_Amount', hue='Gender', ax=axes[1])
    axes[1].set_title('Density Plot of Purchase Amount by Gender')
    axes[1].set_xlabel('Purchase Amount (USD)')
    axes[1].set_ylabel('Density')

    plt.tight_layout()
    plt.show()

display(Markdown('''
The figures above illustrates the differences in browsing and purchasing␣
 ↪behavior between male and female customers
using a density plot for each behavior.  In terms of browsing time, males tend␣
 ↪to spend slightly less time on the platform,
with a higher density around the 20-40 minute range, suggesting more direct or␣
 ↪goal-driven behavior. In contrast,
females exhibit a broader distribution with a secondary peak beyond 60 minutes,␣
 ↪indicating more varied and potentially exploratory browsing patterns.
For purchase amounts, both genders show peaks within the 100-300 range, but␣
 ↪males demonstrate a more dispersed pattern-suggesting a tendency
toward both lower and higher spending extremes. Females, on the other hand,␣
 ↪tend to cluster around mid-range purchase amounts,
showing more consistent spending behavior. This shows that male customers may␣
 ↪respond better to fast, targeted purchase prompts or flexible pricing,
while female customers may benefit more from extended engagement strategies and␣
 ↪mid-range value propositions.
'''))
```



The figures above illustrates the differences in browsing and purchasing behavior between male and female customers using a density plot for each behavior. In terms of browsing time, males tend to spend slightly less time on the platform, with a higher density around the 20–40 minute range, suggesting more direct or goal-driven behavior. In contrast, females exhibit a broader distribution with a secondary peak beyond 60 minutes, indicating more varied and potentially exploratory browsing patterns. For purchase amounts, both genders show peaks within the 100–300 range, but males demonstrate a more dispersed pattern—suggesting a tendency toward both lower and higher spending extremes. Females, on the other hand, tend to cluster around mid-range purchase amounts, showing more consistent spending behavior. This shows that male customers may respond better to fast, targeted purchase prompts or flexible pricing, while female customers may benefit more from extended engagement strategies and mid-range value propositions.

2.4 D. Square Root Transformation on Browsing Time

[13]:
```python
from scipy.stats import skew

# Original skewness of Browsing_Time
original_skew = skew(df['Browsing_Time'])

# Data Frame Copy
df_transformed = df.copy()
df_transformed.drop('Browsing_Time', axis=1, inplace=True) # Drop Browsing_Time␣
↪column

# Applying square root transformation
df_transformed['Browsing_Time'] = np.sqrt(df['Browsing_Time'])

# Skewness after transformation
sqrt_skew  = skew(df_transformed['Browsing_Time'])

# Combine into a DataFrame
skew_comparison = pd.DataFrame({
    'Transformation': ['Original', 'Transformed'],
    'Skewness': [original_skew, sqrt_skew]
})

skew_comparison
```

[13]:

	Transformation	Skewness
0	Original	0.038635
1	Transformed	-0.477074

Evaluation:

- The skewness shifted by approximately −0.516, moving the distribution from slightly right-skewed to moderately left-skewed.
- This indicates that while the original variable was nearly symmetric, applying the square root transformation tilted the distribution left, reducing the influence of high browsing times.

[14]:
```python
df_transformed.describe(include='all')
```

```
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1458:
RuntimeWarning: invalid value encountered in greater
  has_large_values = (abs_vals > 1e6).any()
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1459:
RuntimeWarning: invalid value encountered in less
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any()
/usr/local/lib/python3.11/dist-packages/pandas/io/formats/format.py:1459:
RuntimeWarning: invalid value encountered in greater
  has_small_values = ((abs_vals < 10 ** (-self.digits)) & (abs_vals > 0)).any()
```

[14]:
```
        Customer_ID Gender          Age  Purchase_Amount  Number_of_Items  \
count   3000.000000   3000  3000.000000      3000.000000      3000.000000
unique          NaN      2          NaN              NaN              NaN
top             NaN   Male          NaN              NaN              NaN
freq            NaN   1517          NaN              NaN              NaN
mean    1500.500000    NaN    43.606000       247.962540         4.989667
std      866.169729    NaN    14.963759       140.875783         2.561200
min        1.000000    NaN    18.000000         5.030000         1.000000
25%      750.750000    NaN    31.000000       128.695000         3.000000
50%     1500.500000    NaN    44.000000       245.090000         5.000000
75%     2250.250000    NaN    57.000000       367.200000         7.000000
max     3000.000000    NaN    69.000000       499.610000         9.000000

        Discount_Applied  Total_Transactions  Category  Satisfaction_Score  \
count        3000.000000         3000.000000      3000         3000.000000
unique               NaN                 NaN         5                 NaN
top                  NaN                 NaN  Clothing                 NaN
freq                 NaN                 NaN       629                 NaN
mean           24.345000           24.683000       NaN            3.066000
std            14.709433           14.214518       NaN            1.402723
min             0.000000            1.000000       NaN            1.000000
25%            12.000000           12.000000       NaN            2.000000
50%            24.000000           24.000000       NaN            3.000000
75%            37.000000           37.000000       NaN            4.000000
max            49.000000           49.000000       NaN            5.000000

        Browsing_Time
count     3000.000000
unique            NaN
top               NaN
freq              NaN
mean         7.320228
std          2.507050
min          1.000000
25%          5.475856
```

```
50%          7.691554
75%          9.451455
max         10.952169
```

2.5 E. Linear Model for the Dataset

[15]:
```python
import statsmodels.api as sm

# Define predictors and target
X = df_transformed[['Browsing_Time']]
y = df_transformed['Purchase_Amount']

# Add constant for intercept
X = sm.add_constant(X)

# Fit the linear regression model
model = sm.OLS(y, X).fit()
model.summary()
```

[15]:

Dep. Variable:	Purchase_Amount	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	-0.000
Method:	Least Squares	**F-statistic:**	0.7329
Date:	Tue, 13 May 2025	**Prob (F-statistic):**	0.392
Time:	14:07:32	**Log-Likelihood:**	-19100.
No. Observations:	3000	**AIC:**	3.820e+04
Df Residuals:	2998	**BIC:**	3.822e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	254.3933	7.940	32.041	0.000	238.825	269.961
Browsing_Time	-0.8785	1.026	-0.856	0.392	-2.890	1.134

Omnibus:	1773.526	**Durbin-Watson:**	2.070
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	168.694
Skew:	0.047	**Prob(JB):**	2.34e-37
Kurtosis:	1.842	**Cond. No.**	24.2

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

2.5.1 Interpretation

- **Dependent Variable**: `Purchase_Amount`
- **Independent Variable**: `Browsing_Time`

Key Results

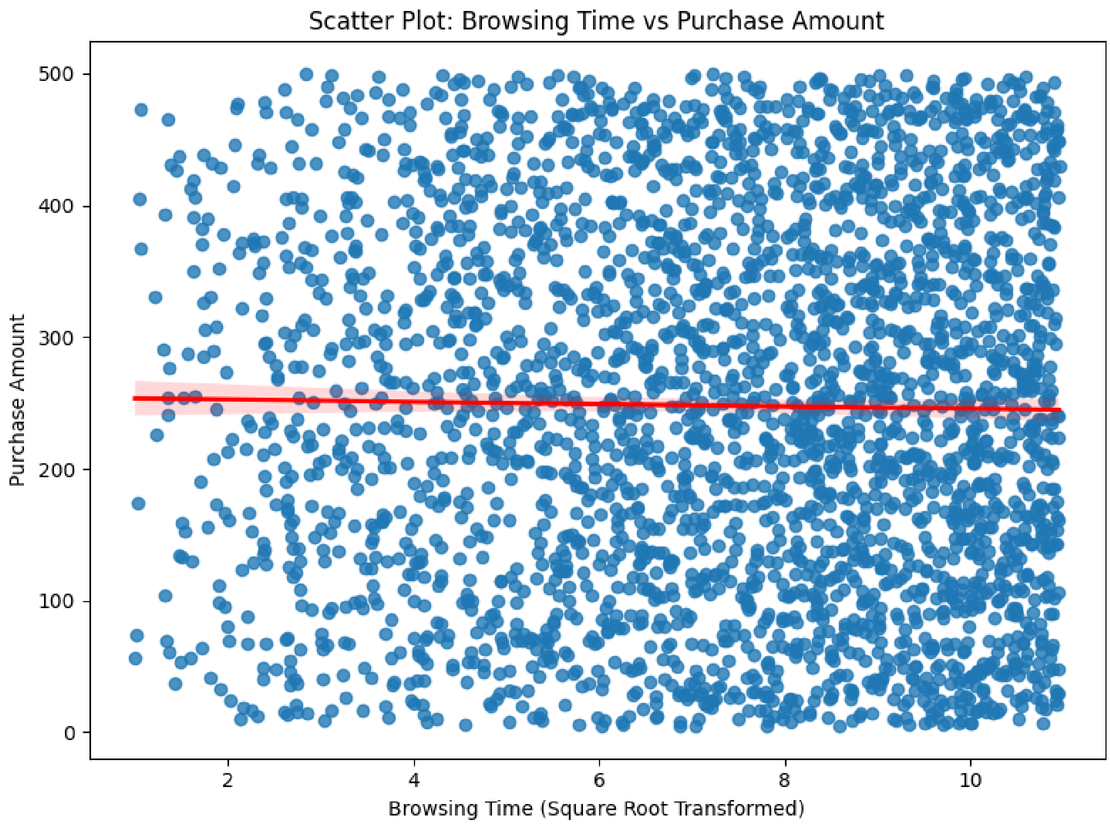
- **Intercept (const)**: 254.39

  - This is the predicted purchase amount when browsing time is 0. It serves as the baseline.
- **`Browsing_Time` coefficient**: -0.88
  - For each unit increase in the square root of browsing time, the purchase amount is predicted to decrease by approximately $0.88. However, this coefficient is not statistically significant.
- **R-squared**: 0.000
  - The model explains virtually none of the variance in purchase amount.
- **p-value for `Browsing_Time`**: 0.392
  - Since this is well above 0.05, we fail to reject the null hypothesis. There is no significant linear relationship between browsing time (even after transformation) and purchase amount.

Conclusion: Browsing time does not significantly predict purchase amount.

2.6 F. Scatter Plot: Browsing Time vs Purchase Amount

[16]:
```
# Create scatter plot with regression line
plt.figure(figsize=(8, 6))
sns.regplot(data=df_transformed, x='Browsing_Time', y='Purchase_Amount',␣
 ↪line_kws={"color": "red"})
plt.title("Scatter Plot: Browsing Time vs Purchase Amount")
plt.xlabel("Browsing Time (Square Root Transformed)")
plt.ylabel("Purchase Amount")
plt.tight_layout()
plt.show()

display(Markdown('''
The figures above illustrates the scatter plot between browsing time and␣
 ↪purchase amount with
a regression line. It shows a very weak and nearly flat negative trend between␣
 ↪the square root
of browsing time and purchase amount. It indicates that there's no strong
or significant linear relationship between the two variables.
'''))
```



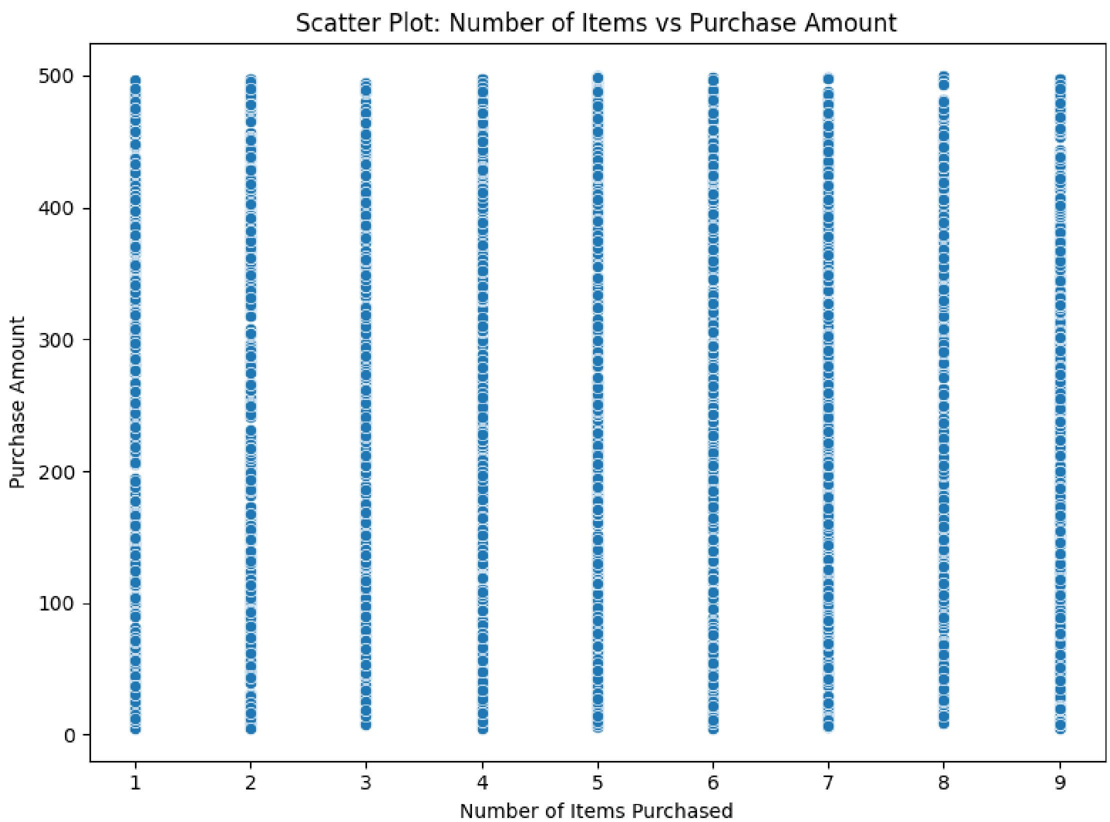
The figures above illustrates the scatter plot between browsing time and purchase amount with a regression line. It shows a very weak and nearly flat negative trend between the square root of browsing time and purchase amount. It indicates that there's no strong or significant linear relationship between the two variables.

3 III. Bivariate Data Analysis

3.1 A. Scatter Plot: Purchase Amount vs Number of Items

[17]:
```
# Scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df_transformed, x='Number_of_Items', y='Purchase_Amount')
plt.title("Scatter Plot: Number of Items vs Purchase Amount")
plt.xlabel("Number of Items Purchased")
plt.ylabel("Purchase Amount")
plt.tight_layout()
plt.show()

display(Markdown('''
#### Interpretation:
```

```
- Each vertical strip of points corresponds to a fixed number of items (ranging␣
 ↪from 1 to 9), with wide variability in the purchase amount.
- There is no strong linear or curvilinear relationship observable from this␣
 ↪plot.
For every item count, purchase amounts are scattered widely from low to high␣
 ↪values.
- This suggests that purchase amount is not strongly determined by the number␣
 ↪of items alone.
Other factors like item category, discounts, or customer behavior might␣
 ↪influence the total amount.
'''))
```



Interpretation:

- Each vertical strip of points corresponds to a fixed number of items (ranging from 1 to 9), with wide variability in the purchase amount.
- There is no strong linear or curvilinear relationship observable from this plot. For every item count, purchase amounts are scattered widely from low to high values.
- This suggests that purchase amount is not strongly determined by the number of items alone. Other factors like item category, discounts, or customer behavior might influence the total amount.

3.2 B. Comparison of Simple Linear and Polynomial Regression Model for Purchase Amount and Browsing Time

[18]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score

# Prepare data for polynomial regression
X_poly = df_transformed[['Browsing_Time']].values
y = df_transformed['Purchase_Amount'].values

# Polynomial features (degree 2)
poly = PolynomialFeatures(degree=2)
X_poly_transformed = poly.fit_transform(X_poly)

# Fit polynomial regression
poly_model = LinearRegression().fit(X_poly_transformed, y)
y_poly_pred = poly_model.predict(X_poly_transformed)

# Fit simple linear regression for comparison
lin_model = LinearRegression().fit(X_poly, y)
y_lin_pred = lin_model.predict(X_poly)

# Compute metrics
poly_mse = mean_squared_error(y, y_poly_pred)
poly_r2 = r2_score(y, y_poly_pred)

lin_mse = mean_squared_error(y, y_lin_pred)
lin_r2 = r2_score(y, y_lin_pred)

# Combine into a DataFrame
model_comparison = pd.DataFrame({
    'Metric': ['Mean Squared Error (MSE)', 'R-squared (R²)'],
    'Simple Linear Model': [lin_mse, lin_r2],
    'Polynomial Regression Model': [poly_mse, poly_r2]
})

model_comparison
```

[18]:

	Metric	Simple Linear Model	Polynomial Regression Model
0	Mean Squared Error (MSE)	19834.521801	19826.446011
1	R-squared (R²)	0.000244	0.000651

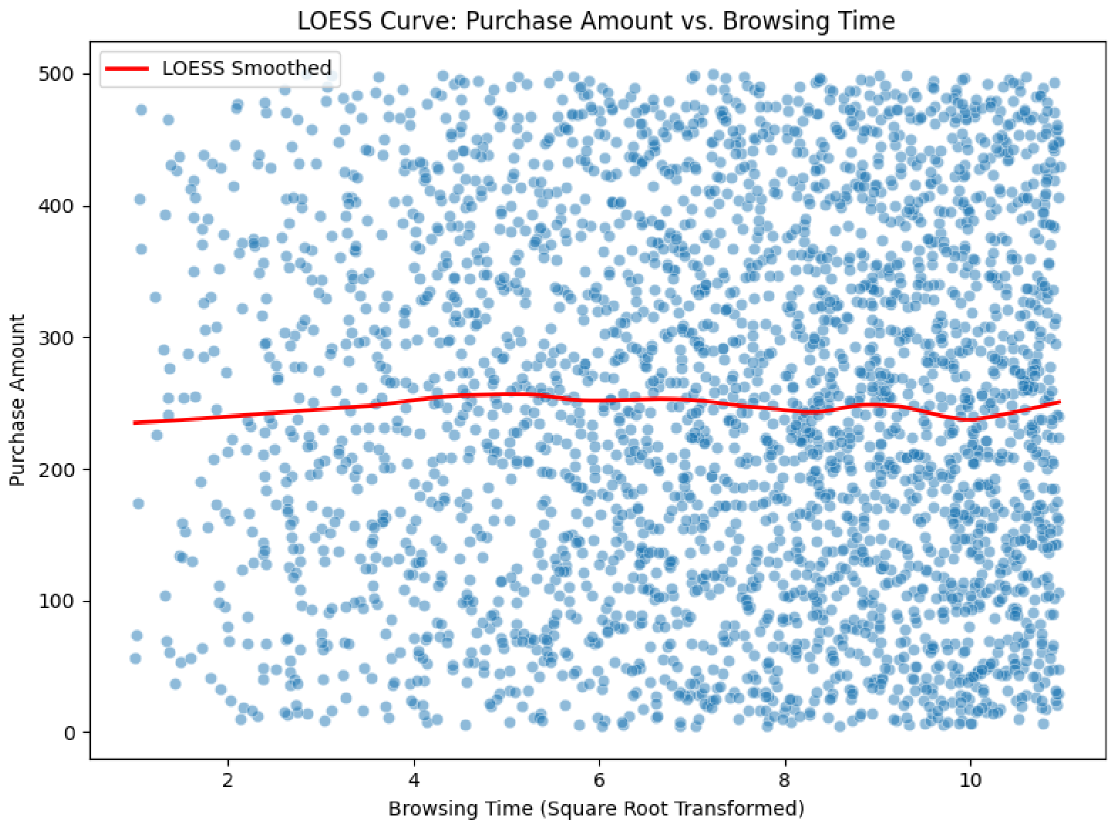
Interpretation:

- Both models perform very poorly, with R^2 values close to zero, indicating almost no explanatory power.
- The polynomial regression shows a slightly lower MSE and higher R^2, but the improvement

is marginal and not practically meaningful.
- This confirms that `Browsing_Time` is not a strong predictor of `Purchase_Amount`, even with a nonlinear (quadratic) model.

3.3 C. Locally Estimated Scatterplot Smoothing: Purchase Amount vs Browsing Time

[19]:
```
import statsmodels.api as sm

# LOESS smoothing
lowess = sm.nonparametric.lowess
loess_smoothed = lowess(df_transformed['Purchase_Amount'],␣
↪df_transformed['Browsing_Time'], frac=0.3)

# Extracting smoothed values
loess_x = loess_smoothed[:, 0]
loess_y = loess_smoothed[:, 1]

# Plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Browsing_Time', y='Purchase_Amount', data=df_transformed,␣
↪alpha=0.5)
plt.plot(loess_x, loess_y, color='red', linewidth=2, label='LOESS Smoothed')
plt.title("LOESS Curve: Purchase Amount vs. Browsing Time")
plt.xlabel("Browsing Time (Square Root Transformed)")
plt.ylabel("Purchase Amount")
plt.legend()
plt.tight_layout()
plt.show()

display(Markdown('''
#### Interpretation:
- The smoothed curve is relatively flat, with only minor undulations.
- This indicates that no strong local patterns or meaningful trends exist␣
↪between browsing time and purchase amount.
- The flatness of the curve supports earlier findings: `Browsing_Time` does not␣
↪significantly influence `Purchase_Amount`, even when analyzed␣
↪non-parametrically.
'''))
```



Interpretation:

- The smoothed curve is relatively flat, with only minor undulations.
- This indicates that no strong local patterns or meaningful trends exist between browsing time and purchase amount.
- The flatness of the curve supports earlier findings: `Browsing_Time` does not significantly influence `Purchase_Amount`, even when analyzed non-parametrically.

3.4 D. Comparison Between Robust Regression Methods and Ordinary Least Squares (OLS)

[20]:
```
import statsmodels.api as sm
from statsmodels.robust.robust_linear_model import RLM
from statsmodels.robust.norms import HuberT, TukeyBiweight

# Data
X = sm.add_constant(df_transformed[['Browsing_Time']])
y = df_transformed['Purchase_Amount']

# OLS model
ols_model = sm.OLS(y, X).fit()
```

```python
# Huber regression
huber_model = RLM(y, X, M=HuberT()).fit()

# Tukey biweight regression
tukey_model = RLM(y, X, M=TukeyBiweight()).fit()

# Data Frame Comparison
comparison = pd.DataFrame({
    'Model': ['OLS', 'Huber', 'Tukey'],
    'Intercept': [ols_model.params[0], huber_model.params[0], tukey_model.
 ↪params[0]],
    'Browsing_Time Coef': [ols_model.params[1], huber_model.params[1],␣
 ↪tukey_model.params[1]],
    'R-squared (OLS)': [ols_model.rsquared, '-', '-']
})

comparison
```

```
/tmp/ipykernel_85/1484777657.py:21: FutureWarning: Series.__getitem__ treating
keys as positions is deprecated. In a future version, integer keys will always
be treated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  'Intercept': [ols_model.params[0], huber_model.params[0],
tukey_model.params[0]],
/tmp/ipykernel_85/1484777657.py:22: FutureWarning: Series.__getitem__ treating
keys as positions is deprecated. In a future version, integer keys will always
be treated as labels (consistent with DataFrame behavior). To access a value by
position, use `ser.iloc[pos]`
  'Browsing_Time Coef': [ols_model.params[1], huber_model.params[1],
tukey_model.params[1]],
```

[20]:

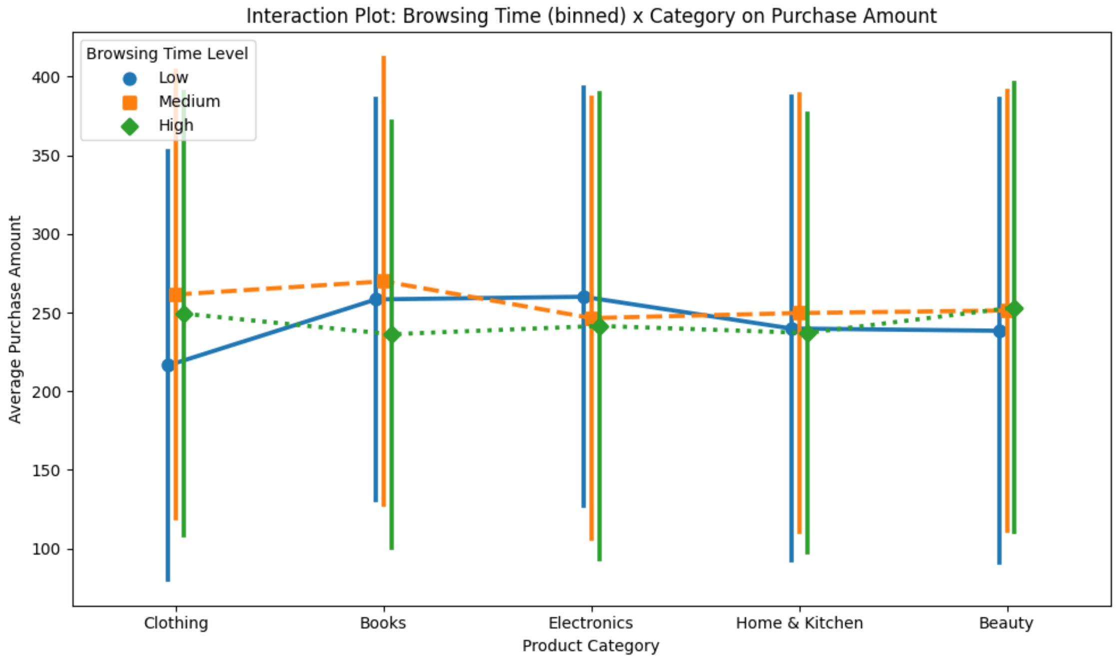
	Model	Intercept	Browsing_Time Coef	R-squared (OLS)
0	OLS	254.393319	-0.878494	0.000244
1	Huber	254.409966	-0.898177	-
2	Tukey	254.859147	-1.001195	-

The table above compares OLS, Huber, and Tukey regression methods. Robust regressions (Huber and Tukey) yield similar coefficients to OLS but are less sensitive to outliers. Although the coefficients differ slightly, the relationship between `Browsing_Time` and `Purchase_Amount` remains weak in all models.

4 IV. Trivariate/Hypervariate Data Analysis

4.1 A. Interaction Effects Between Browsing Time and Category on Purchase Amount

[21]:
```
plt.figure(figsize=(10, 6))
sns.pointplot(
    data=df_transformed,
    x='Category',
    y='Purchase_Amount',
    hue=pd.cut(df_transformed['Browsing_Time'], bins=3, labels=['Low',␣
 ↪'Medium', 'High']),
    dodge=True,
    errorbar='sd',
    markers=['o', 's', 'D'],
    linestyles=['-', '--', ':']
)
plt.title('Interaction Plot: Browsing Time (binned) x Category on Purchase␣
 ↪Amount')
plt.xlabel('Product Category')
plt.ylabel('Average Purchase Amount')
plt.legend(title='Browsing Time Level')
plt.tight_layout()
plt.show()

display(Markdown('''
#### Interpretation:
- For some categories like Books and Electronics, higher browsing time (Medium/
 ↪High) corresponds to slightly lower purchase amounts.
- In Clothing, Medium browsing time users tend to spend more than those with␣
 ↪Low or High.
- There is no consistent trend across all categories, suggesting that the␣
 ↪interaction effect is non-linear and category-specific.
- Overall, the relationship between browsing time and purchase amount is weak␣
 ↪and likely moderated by other variables.
'''))
```



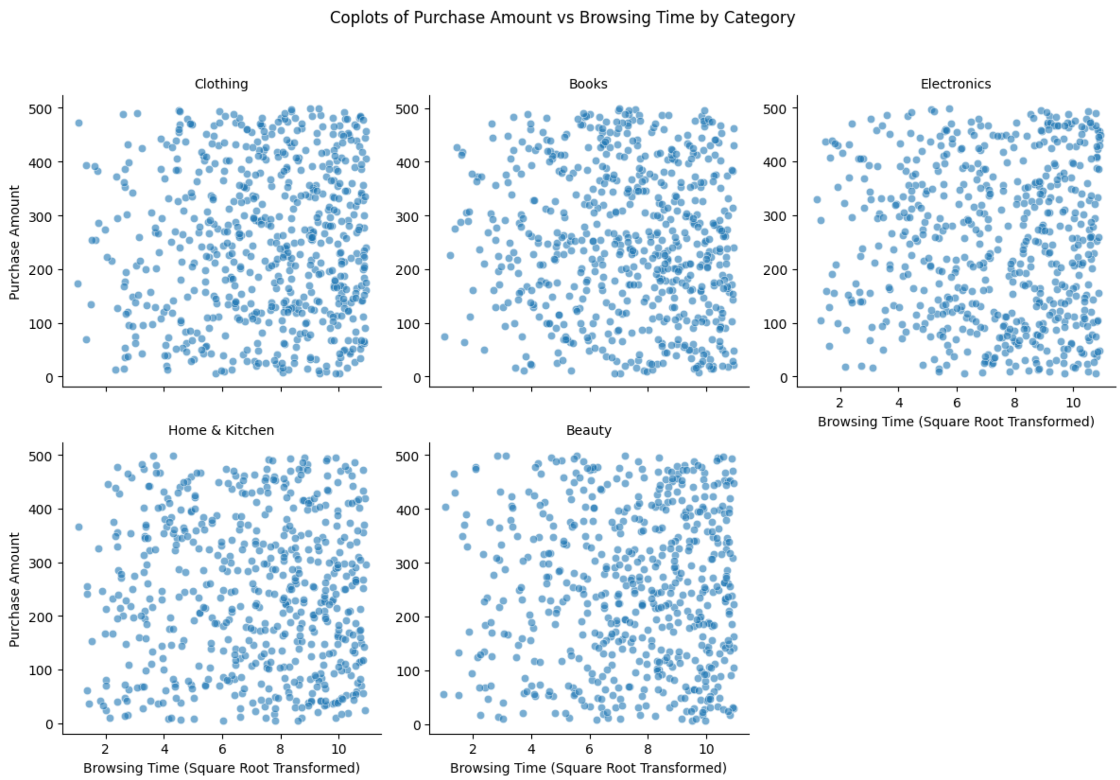
Interpretation:

- For some categories like Books and Electronics, higher browsing time (Medium/High) corresponds to slightly lower purchase amounts.
- In Clothing, Medium browsing time users tend to spend more than those with Low or High.
- There is no consistent trend across all categories, suggesting that the interaction effect is non-linear and category-specific.
- Overall, the relationship between browsing time and purchase amount is weak and likely moderated by other variables.

4.2 B. Coplots of Purchase Amount Against Browsing Time

[22]:
```
g = sns.FacetGrid(df_transformed, col="Category", col_wrap=3, height=4,␣
↪sharey=False)
g.map_dataframe(sns.scatterplot, x="Browsing_Time", y="Purchase_Amount",␣
↪alpha=0.6)
g.set_axis_labels("Browsing Time (Square Root Transformed)", "Purchase Amount")
g.set_titles(col_template="{col_name}")
g.fig.suptitle("Coplots of Purchase Amount vs Browsing Time by Category", y=1.
↪03)
plt.tight_layout()
plt.show()

display(Markdown('''
#### Interpretation:
```

```
- Across most categories, no strong trend or pattern emerges. Points remain␣
↪widely scattered.
- In Books and Home & Kitchen, some slight downward trends appear with␣
↪increasing browsing time, but not consistently.
- Clothing and Electronics show broad dispersion, suggesting high variability␣
↪regardless of browsing behavior.
- In conclusion, these coplots reinforce earlier findings - the relationship␣
↪between browsing time and purchase amount is weak and likely moderated by␣
↪other variables.
'''))
```

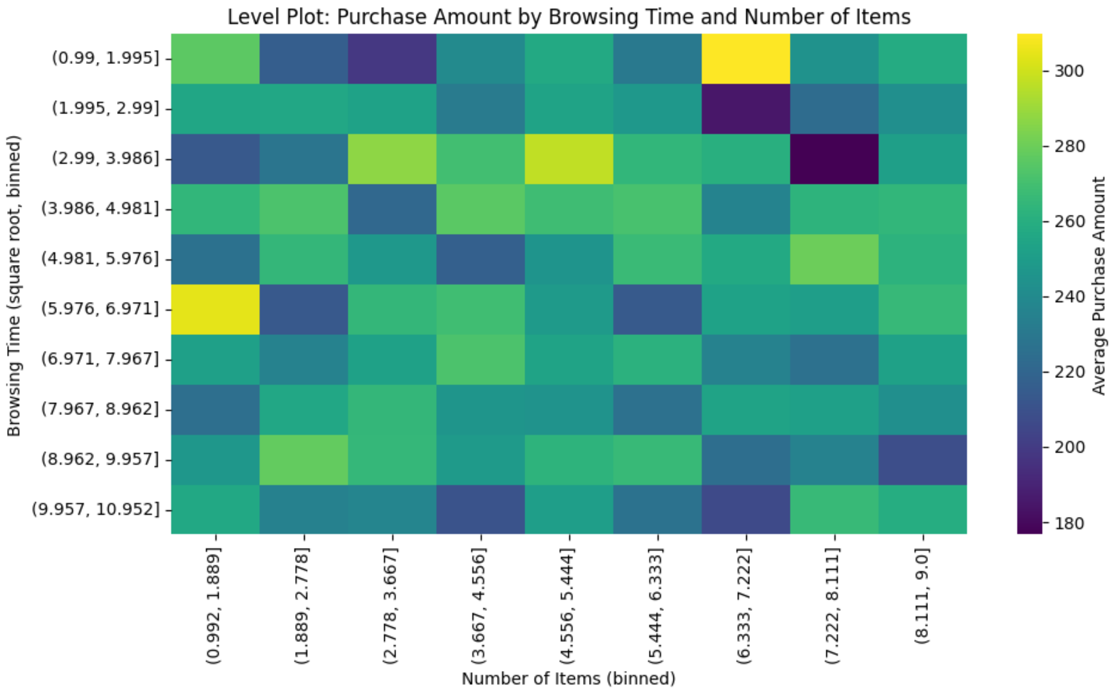


Interpretation:

- Across most categories, no strong trend or pattern emerges. Points remain widely scattered.
- In Books and Home & Kitchen, some slight downward trends appear with increasing browsing time, but not consistently.
- Clothing and Electronics show broad dispersion, suggesting high variability regardless of browsing behavior.
- In conclusion, these coplots reinforce earlier findings — the relationship between browsing time and purchase amount is weak and likely moderated by other variables.

4.3 C. Relationship Visualization: Browsing Time, Number of Items, and Purchase Amount

[23]:
```
# Level plot
pivot_table = df_transformed.pivot_table(
    values='Purchase_Amount',
    index=pd.cut(df_transformed['Browsing_Time'], bins=10),
    columns=pd.cut(df_transformed['Number_of_Items'], bins=9),
    aggfunc='mean',
    observed=False
)

# Plot
plt.figure(figsize=(10, 6))
sns.heatmap(pivot_table, annot=False, cmap='viridis', cbar_kws={'label':␣
 ↪'Average Purchase Amount'})
plt.title('Level Plot: Purchase Amount by Browsing Time and Number of Items')
plt.xlabel('Number of Items (binned)')
plt.ylabel('Browsing Time (square root, binned)')
plt.tight_layout()
plt.show()

display(Markdown('''
#### Interpretation:
- Darker areas represent lower purchase amounts, while lighter areas indicate␣
 ↪higher spending.
- The highest purchase amounts are concentrated in the upper-right region,␣
 ↪suggesting that customers who spend more time browsing and purchase more␣
 ↪items tend to spend more overall.
- However, there are some mid-range browsing bins where purchase amounts are␣
 ↪still moderate, indicating some variability.
'''))
```



Interpretation:

- Darker areas represent lower purchase amounts, while lighter areas indicate higher spending.
- The highest purchase amounts are concentrated in the upper-right region, suggesting that customers who spend more time browsing and purchase more items tend to spend more overall.
- However, there are some mid-range browsing bins where purchase amounts are still moderate, indicating some variability.

4.4 D. Multiple Linear Regression Model

[24]:

```
# Data
X_multi = df_transformed[['Browsing_Time', 'Number_of_Items',␣
↪'Satisfaction_Score']]
y = df_transformed['Purchase_Amount']

# Constant for intercept
X_multi = sm.add_constant(X_multi)

# Fiting the multiple regression model
multi_model = sm.OLS(y, X_multi).fit()
multi_model.summary()
```

[24]:

Dep. Variable:	Purchase_Amount	**R-squared:**	0.001
Model:	OLS	**Adj. R-squared:**	-0.000
Method:	Least Squares	**F-statistic:**	0.6725
Date:	Tue, 13 May 2025	**Prob (F-statistic):**	0.569
Time:	14:07:35	**Log-Likelihood:**	-19099.
No. Observations:	3000	**AIC:**	3.821e+04
Df Residuals:	2996	**BIC:**	3.823e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	263.1461	11.077	23.757	0.000	241.428	284.865
Browsing_Time	-0.8966	1.027	-0.873	0.383	-2.910	1.117
Number_of_Items	-0.7832	1.005	-0.779	0.436	-2.754	1.188
Satisfaction_Score	-1.5370	1.835	-0.838	0.402	-5.134	2.060

Omnibus:	1759.018	**Durbin-Watson:**	2.071
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	168.394
Skew:	0.047	**Prob(JB):**	2.71e-37
Kurtosis:	1.843	**Cond. No.**	42.2

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

4.4.1 Interpretation

- **Dependent Variable**: `Purchase_Amount`
- **Independent Variable**: `Browsing_Time`, `Number_of_Items`, and `Satisfaction_Score`

Key Results

- **Intercept (const)**: 263.15
  - Baseline purchase amount when all predictors are 0
- **`Browsing_Time` coefficient**: -0.9
  - A p-value of 0.383 indicates that this coefficient is not statistically significant.
- **`Browsing_Time` coefficient**: -0.78
  - A p-value of 0.436 indicates that this coefficient is not statistically significant.
- **`Satisfaction_Score` coefficient**: -1.54
  - A p-value of 0.402 indicates that this coefficient is not statistically significant.
- **R-squared**: 0.001
  - The model explains only 0.1% of the variance in `Purchase_Amount`.
- **F-statistic p-value**: 0.569
  - Overall model is not significant.

Conclusion: None of the predictors—Browsing Time, Number of Items, or Satisfaction Score—significantly explain variation in Purchase Amount. This implies: - The outcome might be influenced by unobserved variables like product price, promotion type, or customer loyalty. - The relationship could be non-linear or vary across subgroups.

5 V. Model Feature Selection

[25]:
```python
from sklearn.feature_selection import RFE
from sklearn.preprocessing import StandardScaler

# Prepare the features and target
features = df_transformed[['Browsing_Time', 'Number_of_Items', 'Satisfaction_Score']]
target = df_transformed['Purchase_Amount']

# Standardize the features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Fit linear model with recursive feature elimination (RFE)
lr = LinearRegression()
rfe = RFE(estimator=lr, n_features_to_select=1)
rfe.fit(features_scaled, target)

# Create a DataFrame with feature ranking
feature_importance = pd.DataFrame({
    'Feature': features.columns,
    'Ranking': rfe.ranking_,
    'Selected': rfe.support_
}).sort_values(by='Ranking')

feature_importance
```

[25]:

	Feature	Ranking	Selected
0	Browsing_Time	1	True
2	Satisfaction_Score	2	False
1	Number_of_Items	3	False

Interpretation:

- `Browsing_Time` was selected as the most important predictor among the three.
- However, as seen from prior regression models, even this top predictor explains very little variance in `Purchase_Amount` (R^2 0).
- Thus, although `Browsing_Time` is the best among the available variables, its predictive power is still weak, suggesting:
  - The model may benefit from additional or alternative predictors.
  - Relationships might be nonlinear or require interaction modeling.