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

# Providing the path in drive
file_path = 'shopping_trends_updated - shopping_trends_updated.csv'

# Loading the data
df = pd.read_csv(file_path)

# Display the first few rows of the dataset
df.head()
```

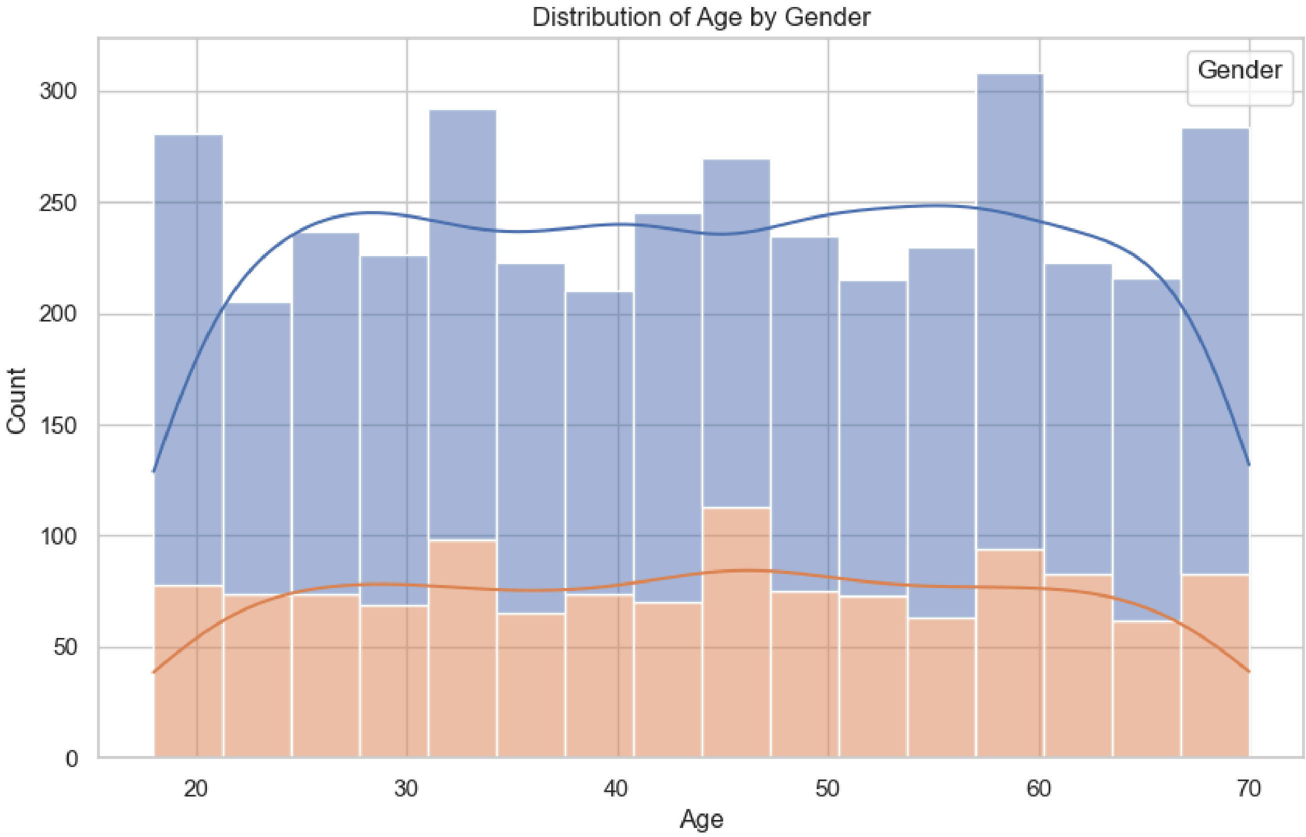
Out[ ]:

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season
0	1	55	Male	Blouse	Clothing	53	Kentucky	L	Gray	Winter
1	2	19	Male	Sweater	Clothing	64	Maine	L	Maroon	Winter
2	3	50	Male	Jeans	Clothing	73	Massachusetts	S	Maroon	Spring
3	4	21	Male	Sandals	Footwear	90	Rhode Island	M	Maroon	Spring
4	5	45	Male	Blouse	Clothing	49	Oregon	M	Turquoise	Spring

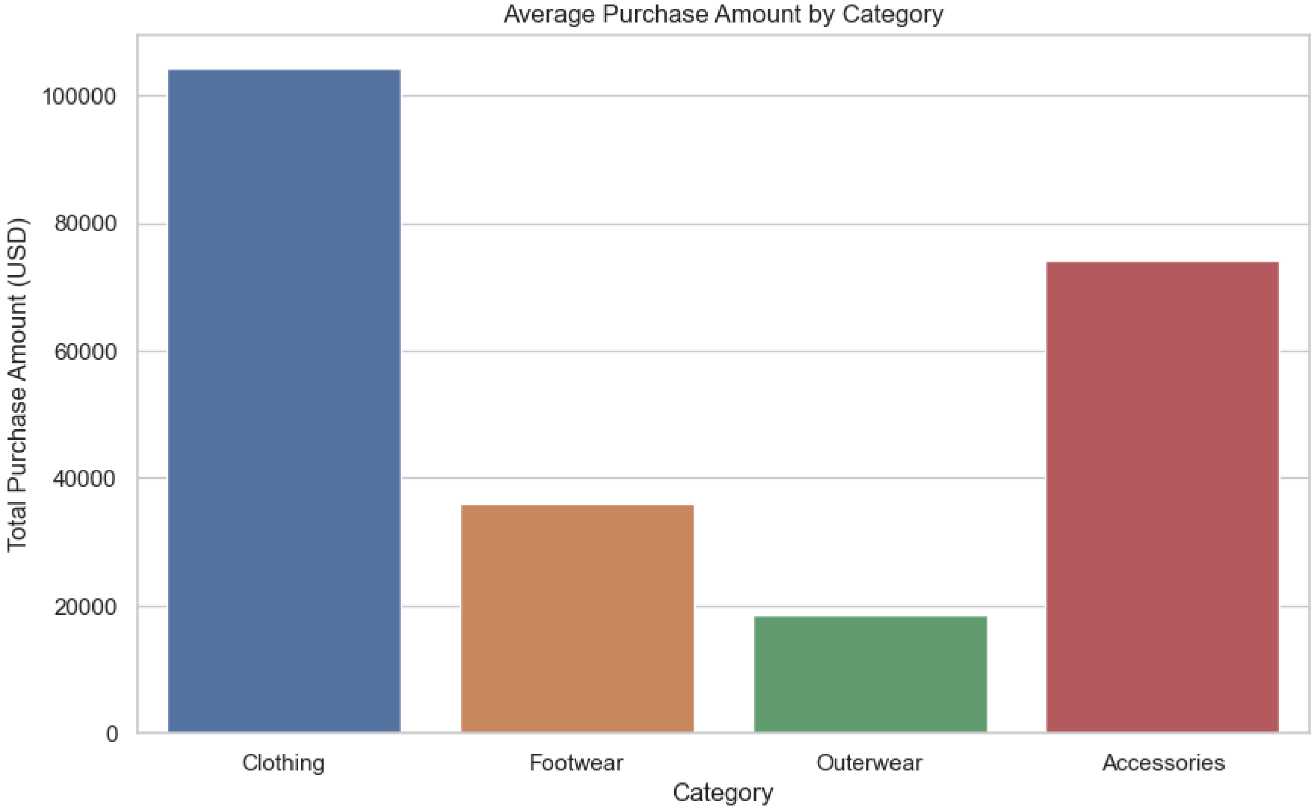
In [ ]:
```
# Set the style for the plots
sns.set(style="whitegrid")

# Plot 1: Distribution of Age by Gender
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Age', hue='Gender', multiple='stack', kde=True)
plt.title('Distribution of Age by Gender')
plt.xlabel('Age')
plt.ylabel('Count')
plt.legend(title='Gender')
plt.show()

# Plot 2: Average Purchase Amount by Category
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='Category', y='Purchase Amount (USD)', ci=None, estimator=sum)
plt.title('Average Purchase Amount by Category')
plt.xlabel('Category')
plt.ylabel('Total Purchase Amount (USD)')
plt.show()

# Plot 3: Review Ratings by Item Purchased
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='Item Purchased', y='Review Rating')
```

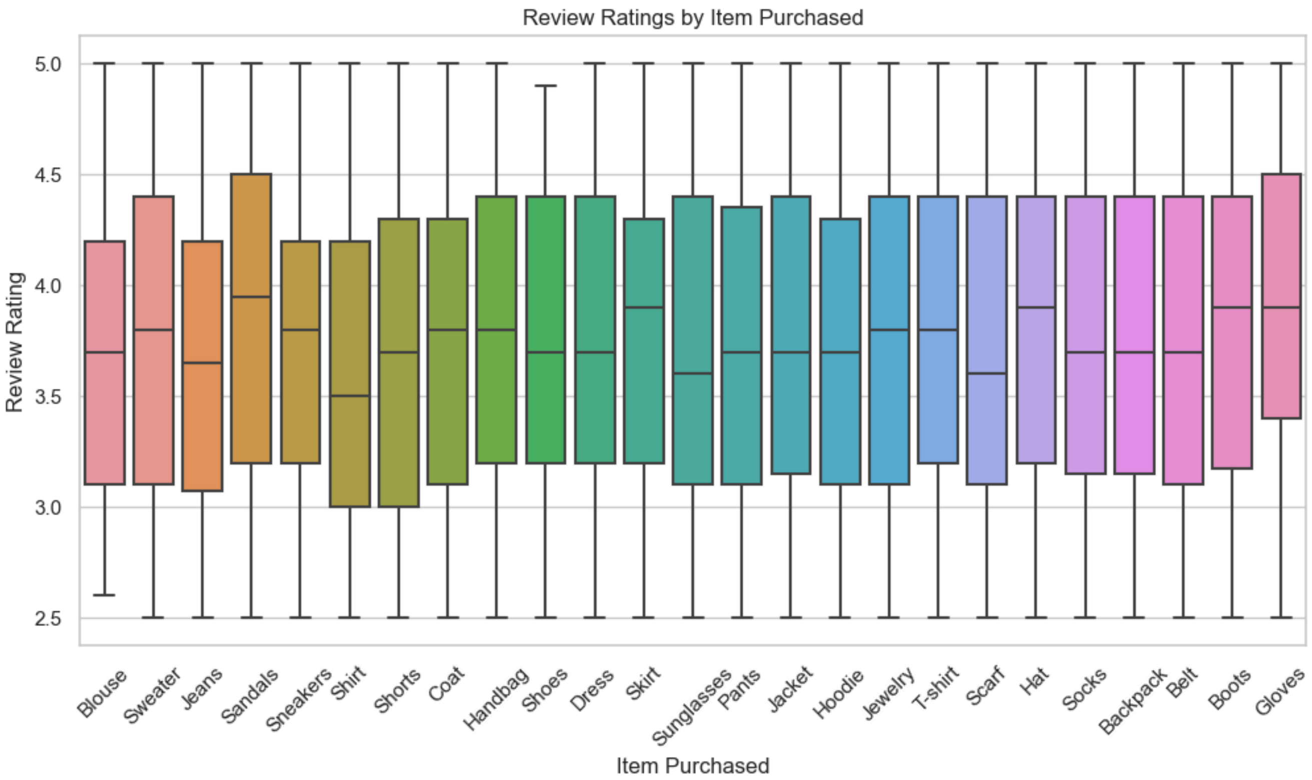
```
plt.title('Review Ratings by Item Purchased')
plt.xlabel('Item Purchased')
plt.ylabel('Review Rating')
plt.xticks(rotation=45)
plt.show()

# Plot 4: Frequency of Purchases by Payment Method
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='Payment Method', hue='Frequency of Purchases')
plt.title('Frequency of Purchases by Payment Method')
plt.xlabel('Payment Method')
plt.ylabel('Count')
plt.legend(title='Frequency of Purchases')
plt.show()

# Plot 5: Seasonal Purchases by Location
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='Location', hue='Season')
plt.title('Seasonal Purchases by Location')
plt.xlabel('Location')
plt.ylabel('Count')
plt.legend(title='Season')
plt.xticks(rotation=45)
plt.show()
```

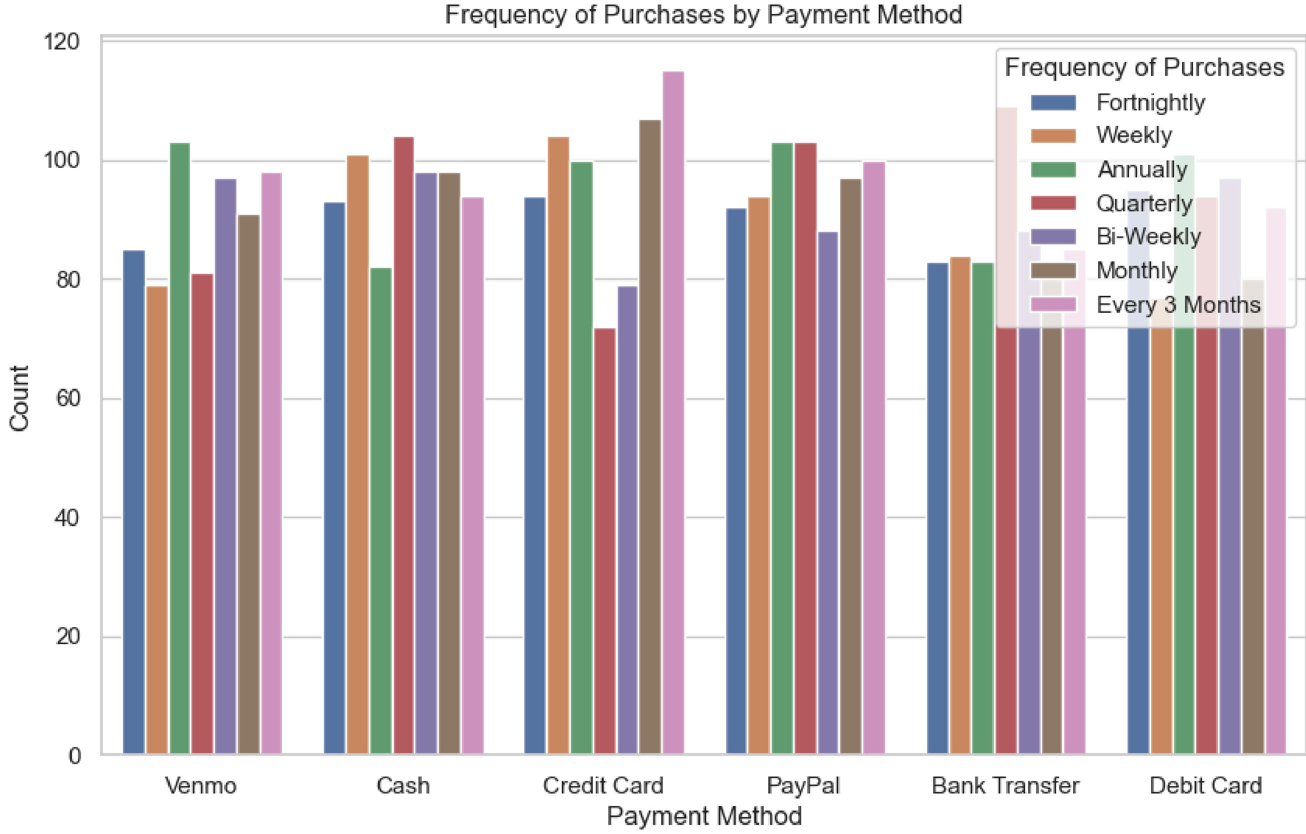
```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
```

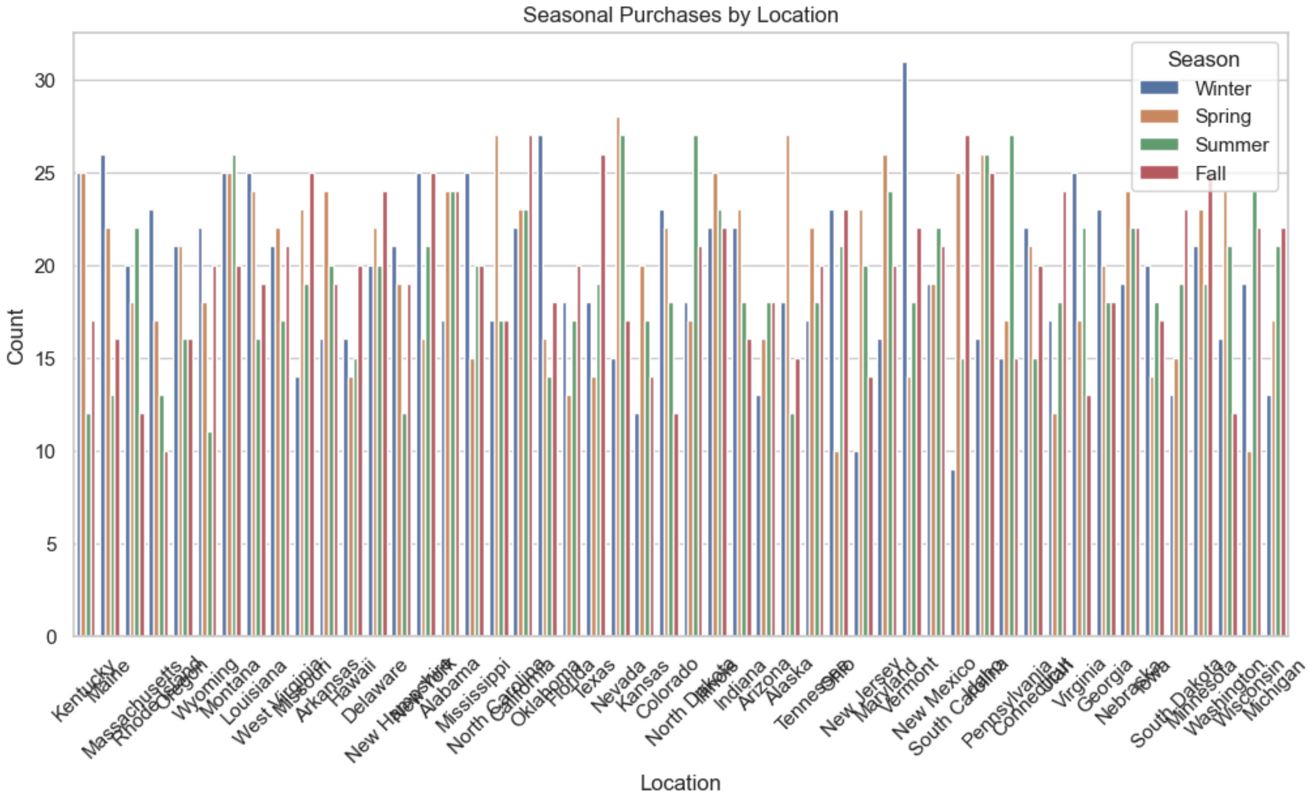


```
C:\Users\aravi\AppData\Local\Temp\ipykernel_13424\3989714833.py:15: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=df, x='Category', y='Purchase Amount (USD)', ci=None, estimator=sum)
```










```
In [ ]: from sklearn.preprocessing import LabelEncoder
        # Check for missing values
        missing_values = df.isnull().sum()

        # Encode all categorical variables using Label encoding
        label_encoders = {}
        for column in df.select_dtypes(include=['object']).columns:
```

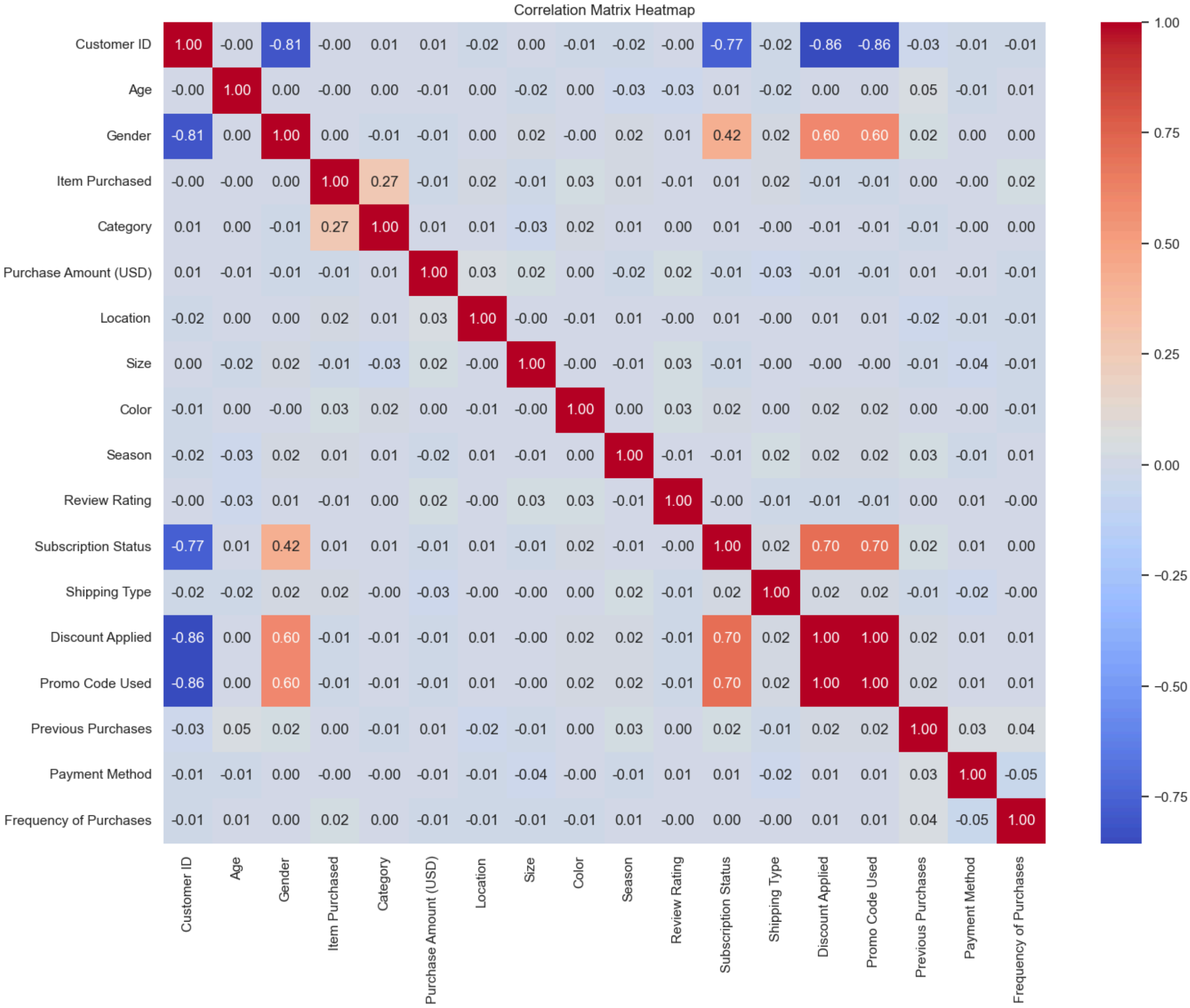
```
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le

# Check for outliers in numeric columns using IQR method
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

# Define outlier criteria
outlier_criteria = (df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))

# Handle outliers: for simplicity, let's remove rows with outliers
df_cleaned = df[~outlier_criteria.any(axis=1)]

# Show the summary of the cleaned dataset
print("Missing values:\n", missing_values)
print("\nCleaned dataset info:\n")
df_cleaned.info()
print("\nPreview of the cleaned dataset:\n", df_cleaned.head())
```

```
Missing values:
 Customer ID              0
Age                      0
Gender                   0
Item Purchased           0
Category                 0
Purchase Amount (USD)    0
Location                 0
Size                     0
Color                    0
Season                   0
Review Rating            0
Subscription Status      0
Shipping Type            0
Discount Applied         0
Promo Code Used          0
Previous Purchases       0
Payment Method           0
Frequency of Purchases   0
dtype: int64

Cleaned dataset info:

<class 'pandas.core.frame.DataFrame'>
Index: 3576 entries, 0 to 3899
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Customer ID             3576 non-null   int64  
 1   Age                     3576 non-null   int64  
 2   Gender                  3576 non-null   int32  
 3   Item Purchased          3576 non-null   int32  
 4   Category                3576 non-null   int32  
 5   Purchase Amount (USD)   3576 non-null   int64  
 6   Location                3576 non-null   int32  
 7   Size                    3576 non-null   int32  
 8   Color                   3576 non-null   int32  
 9   Season                  3576 non-null   int32  
 10  Review Rating           3576 non-null   float64
 11  Subscription Status     3576 non-null   int32  
 12  Shipping Type           3576 non-null   int32  
 13  Discount Applied        3576 non-null   int32  
 14  Promo Code Used         3576 non-null   int32  
 15  Previous Purchases      3576 non-null   int64  
 16  Payment Method          3576 non-null   int32  
 17  Frequency of Purchases  3576 non-null   int32  
dtypes: float64(1), int32(13), int64(4)
memory usage: 349.2 KB

Preview of the cleaned dataset:
    Customer ID  Age  Gender  Item Purchased  Category  Purchase Amount (USD)  \
0            1   55       1               2         1                     53   
1            2   19       1              23         1                     64   
2            3   50       1              11         1                     73   
3            4   21       1              14         2                     90   
4            5   45       1               2         1                     49   
```

```
   Location  Size  Color  Season  Review Rating  Subscription Status  \
0        16     0      7       3            3.1                    1
1        18     0     12       3            3.1                    1
2        20     2     12       1            3.1                    1
3        38     1     12       1            3.5                    1
4        36     1     21       1            2.7                    1

   Shipping Type  Discount Applied  Promo Code Used  Previous Purchases  \
0              1                 1                1                  14
1              1                 1                1                   2
2              2                 1                1                  23
3              3                 1                1                  49
4              2                 1                1                  31

   Payment Method  Frequency of Purchases
0               5                       3
1               1                       3
2               2                       6
3               4                       6
4               4                       0
```

In [ ]:

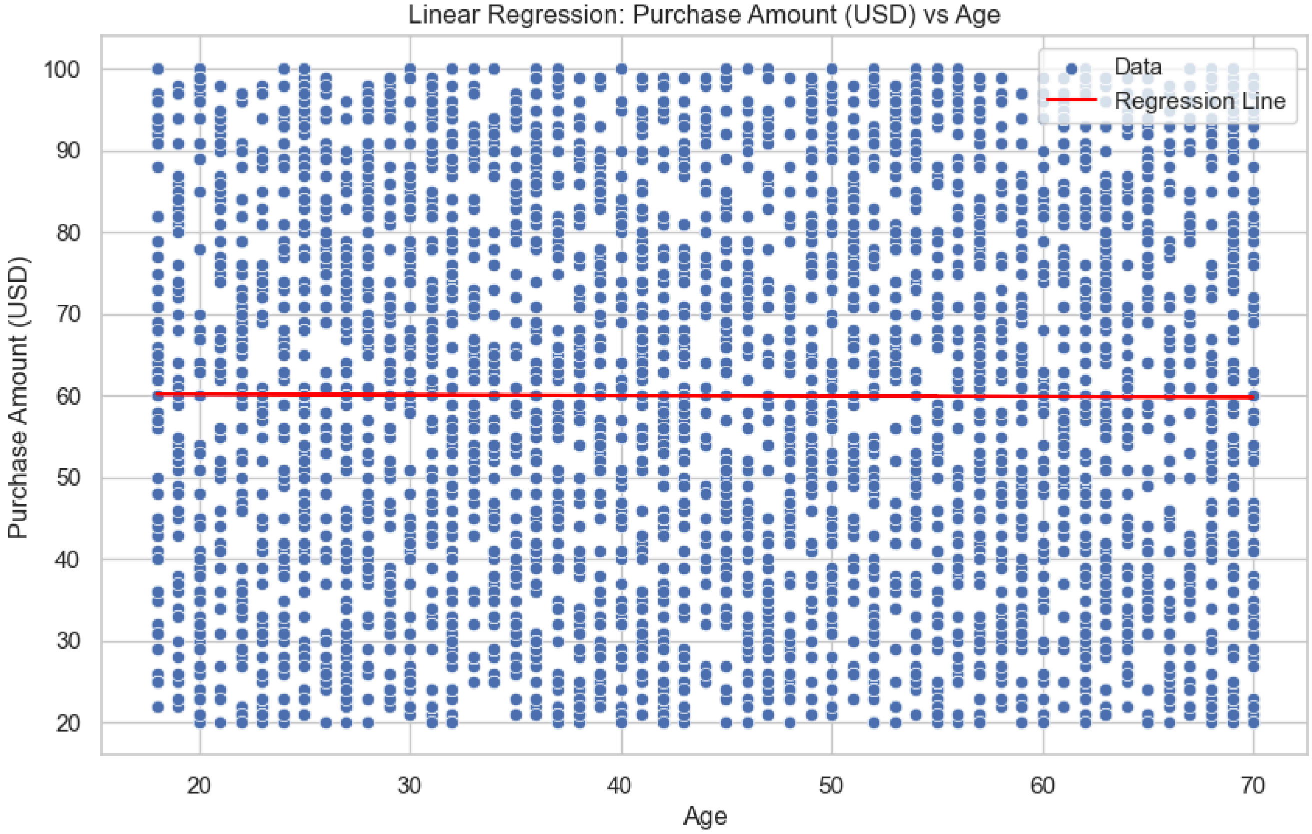
```python
# Calculate the correlation matrix
correlation_matrix = df_cleaned.corr()

# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(16, 12))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", cbar=True)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



In [ ]:
```
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression

X = df_cleaned[['Age']]
y = df_cleaned['Purchase Amount (USD)']

model = LinearRegression()
model.fit(X, y)

# Interpret the regression coefficients
intercept = model.intercept_
slope = model.coef_[0]

print(f'Intercept: {intercept}')
print(f'Slope: {slope}')

# Determine the goodness-of-fit (R-squared value)
y_pred = model.predict(X)
r_squared = r2_score(y, y_pred)
print(f'R-squared: {r_squared}')

# Visualize the regression line on the scatter plot
plt.figure(figsize=(10, 6))
```

```
sns.scatterplot(x='Age', y='Purchase Amount (USD)', data=df_cleaned, label='Data')
plt.plot(X, y_pred, color='red', label='Regression Line')
plt.xlabel('Age')
plt.ylabel('Purchase Amount (USD)')
plt.title('Linear Regression: Purchase Amount (USD) vs Age')
plt.legend()
plt.show()

# Make predictions for new ages 25, 35, and 45
new_ages = np.array([[25], [35], [45]])
predictions = model.predict(new_ages)
print(f'Predicted Purchase Amounts for ages 25, 35, and 45: {predictions}')
```

```
Intercept: 60.35352215281485
Slope: -0.008045173122124808
R-squared: 2.6848368337106798e-05
```




```
Predicted Purchase Amounts for ages 25, 35, and 45: [60.15239282 60.07194109 59.99148936]
```

```
C:\Users\aravi\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [ ]:

```
# • Fit a multiple linear regression model to predict Purchase Amount (USD)
# based on Age, Gender, Location, and Review Rating.
# • Interpret the regression coefficients and their significance.
# • Determine the goodness-of-fit (R-squared value).
# • Make predictions using the regression model for specific values of the
# predictors.

# Define the dependent and independent variables
X = df_cleaned[['Age', 'Gender', 'Location', 'Review Rating']]
y = df_cleaned['Purchase Amount (USD)']
```

```
# Fit the multiple linear regression model
model = LinearRegression()
model.fit(X, y)

# Get the regression coefficients and intercept
coefficients = model.coef_
intercept = model.intercept_

# Print the regression coefficients and intercept
print('Regression coefficients:', coefficients)
print('Intercept:', intercept)

# Determine the goodness-of-fit (R-squared value)
y_pred = model.predict(X)
r_squared = r2_score(y, y_pred)
print('R-squared:', r2_score(y, y_pred))

# Make predictions for specific values of the predictors
new_data = [[30, 1, 2, 4], [45, 0, 1, 3]]
predictions = model.predict(new_data)
print('Predicted purchase amounts:', predictions)

# Check the significance of the regression coefficients using a t-test
import statsmodels.api as sm

X_with_constant = sm.add_constant(X)
model_sm = sm.OLS(y, X_with_constant).fit()
print(model_sm.summary())
```

```
C:\Users\aravi\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

```
Regression coefficients: [-0.00695873 -0.62055956  0.05076208  0.80223492]
Intercept: 56.48461931816324
R-squared: 0.0017198607777986208
Predicted purchase amounts: [58.96576169 58.62894331]
                              OLS Regression Results
=================================================================================
Dep. Variable:     Purchase Amount (USD)   R-squared:                       0.002
Model:                               OLS   Adj. R-squared:                  0.001
Method:                    Least Squares   F-statistic:                     1.538
Date:                   Mon, 24 Jun 2024   Prob (F-statistic):              0.188
Time:                           16:00:03   Log-Likelihood:                -16374.
No. Observations:                   3576   AIC:                         3.276e+04
Df Residuals:                       3571   BIC:                         3.279e+04
Df Model:                              4
Covariance Type:               nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const            56.4846      2.570     21.978      0.000      51.446      61.524
Age              -0.0070      0.026     -0.268      0.789      -0.058       0.044
Gender           -0.6206      0.845     -0.734      0.463      -2.277       1.036
Location          0.0508      0.027      1.849      0.065      -0.003       0.105
Review Rating     0.8022      0.550      1.459      0.145      -0.276       1.881
==============================================================================
Omnibus:                     3673.807   Durbin-Watson:                   1.944
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              222.359
Skew:                          -0.002   Prob(JB):                     5.19e-49
Kurtosis:                       1.778   Cond. No.                         349.
==============================================================================

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

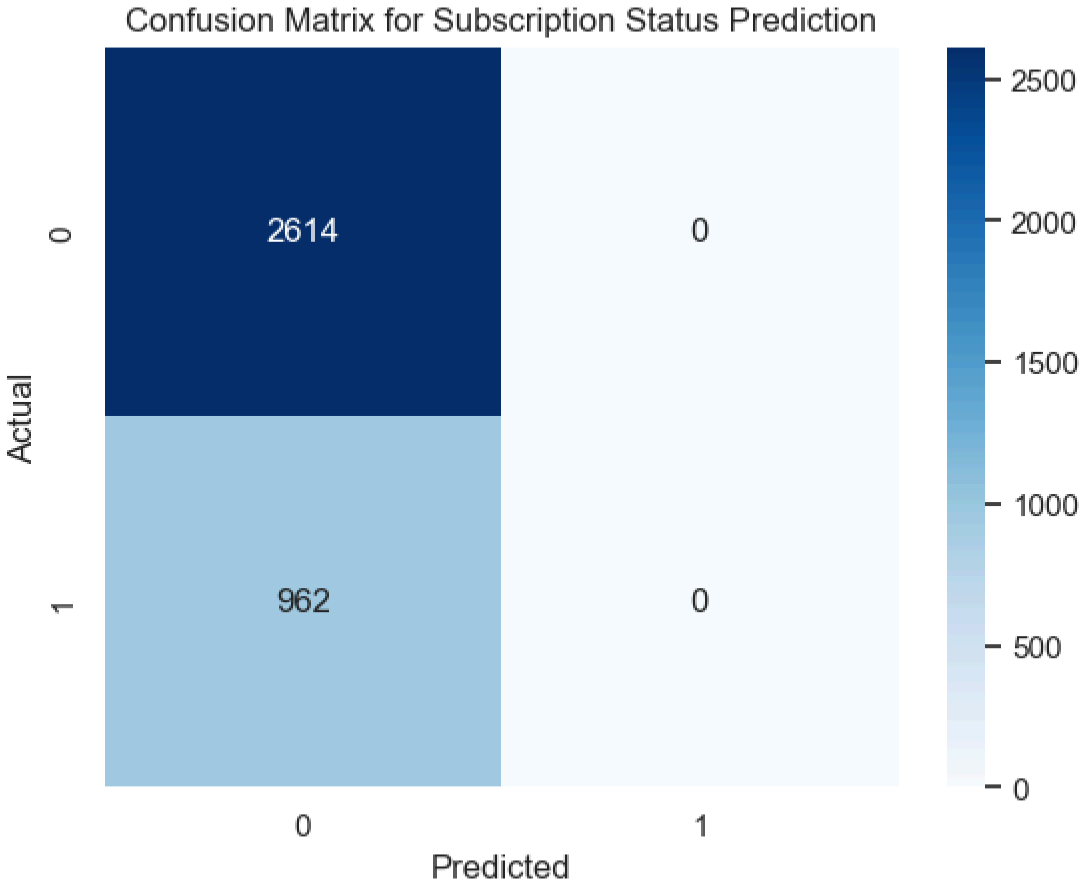
In [ ]:

```python
from sklearn.linear_model import LogisticRegression # Import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confus
import seaborn as sns
# Define the dependent and independent variables
X = df_cleaned[['Age', 'Gender', 'Review Rating']]
y = df_cleaned['Subscription Status']

# Fit the logistic regression model
model = LogisticRegression()
model.fit(X, y)

# Get the regression coefficients and intercept
coefficients = model.coef_
intercept = model.intercept_

# Print the regression coefficients and intercept
print('Regression coefficients:', coefficients)
print('Intercept:', intercept)

# Make predictions for specific values of the predictors
new_data = [[30, 1, 4], [45, 0, 3]]
predictions = model.predict(new_data)
print('Predicted subscription status:', predictions)
```

```
# Evaluate the model using accuracy, precision, recall, and F1-score
y_pred = model.predict(X)
accuracy = accuracy_score(y, y_pred)
precision = precision_score(y, y_pred)
recall = recall_score(y, y_pred)
f1_score = f1_score(y, y_pred)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1-score:', f1_score)

# Create a confusion matrix to visualize the model's performance
confusion_matrix = confusion_matrix(y, y_pred)
sns.heatmap(confusion_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix for Subscription Status Prediction')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
Regression coefficients: [[ 1.42181364e-03  5.00078002e+00 -1.95295013e-02]]
Intercept: [-5.4207814]
Predicted subscription status: [0 0]
Accuracy: 0.7309843400447428
Precision: 0.0
Recall: 0.0
F1-score: 0.0
```

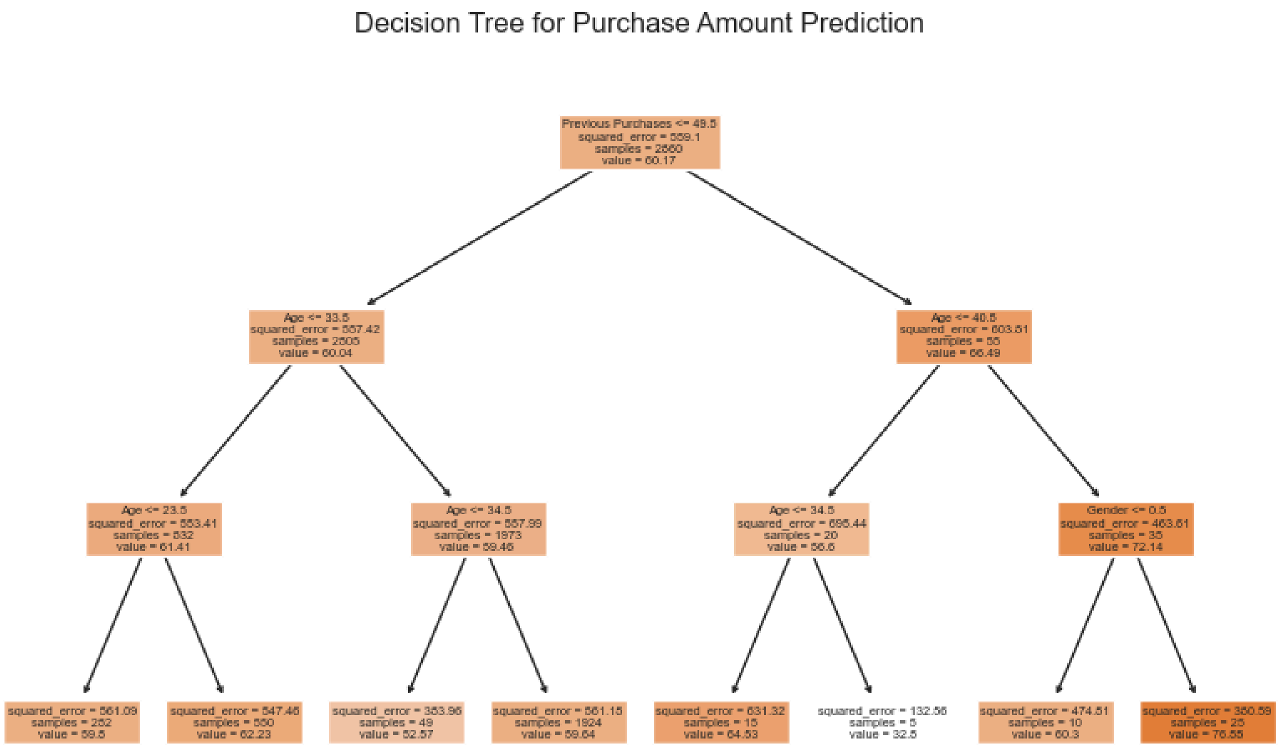
```
C:\Users\aravi\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
C:\Users\aravi\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\metrics\_classification.py:1497: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



In [ ]:
```
# prompt: Decision Tree Regression
# • Fit a decision tree regression model to predict Purchase Amount (USD)
# based on Age, Gender, and Previous Purchases.
# • Visualize the decision tree.
# • Evaluate the model using metrics such as Mean Absolute Error (MAE), Mean
# Squared Error (MSE), and R-squared.
# • Compare the decision tree model with a multiple linear regression model.

import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Define the dependent and independent variables
X = df_cleaned[['Age', 'Gender', 'Previous Purchases']]
y = df_cleaned['Purchase Amount (USD)']

# Split the data into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Fit the decision tree regression model
model =  DecisionTreeRegressor(max_depth=3, random_state=42)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model using MAE, MSE, and R-squared
mae = mean_absolute_error(y_test, y_pred)
```

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

print('Decision Tree Regression:')
print(f'MAE: {mae}')
print(f'MSE: {mse}')
print(f'R-squared: {r2}')

# Visualize the decision tree
from sklearn.tree import plot_tree
plt.figure(figsize=(10, 6))
plot_tree(model, feature_names=X.columns, filled=True, precision=2)
plt.title('Decision Tree for Purchase Amount Prediction')
plt.show()

# Compare the decision tree model with a multiple linear regression model
X_train_with_constant = sm.add_constant(X_train)
model_sm = sm.OLS(y_train, X_train_with_constant).fit()
y_pred_linear = model_sm.predict(sm.add_constant(X_test))

mae_linear = mean_absolute_error(y_test, y_pred_linear)
mse_linear = mean_squared_error(y_test, y_pred_linear)
r2_linear = r2_score(y_test, y_pred_linear)

print('Multiple Linear Regression:')
print(f'MAE: {mae_linear}')
print(f'MSE: {mse_linear}')
print(f'R-squared: {r2_linear}')
```

```
Decision Tree Regression:
MAE: 20.244116169199966
MSE: 551.6320392972503
R-squared: -0.012119022902776555
```



```
Multiple Linear Regression:
MAE: 20.24106704027441
MSE: 546.7865152388188
R-squared: -0.003228591009590165
```