Step 1: Import Necessary Libraries

Start by importing the essential Python libraries.

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

Step 2: Load the Dataset

Load the dataset(s) into a pandas DataFrame.

```
# Load datasets
customers = pd.read_csv('Customers.csv')
products = pd.read_csv('Products.csv')
transactions = pd.read_csv('Transactions.csv')

# Preview the datasets
print(customers.head())
print(products.head())
print(transactions.head())
```

Step 3: Inspect the Data

Understand the structure and contents of the datasets.

```
# Check the shape of the datasets
print("Customers shape:", customers.shape)
print("Products shape:", products.shape)
print("Transactions shape:", transactions.shape)

# Check data types and null values
print(customers.info())
print(products.info())
print(transactions.info())

# Check for missing values
print(customers.isnull().sum())
print(products.isnull().sum())
print(transactions.isnull().sum())
```

Step 4: Summarize the Data

Get a statistical summary of numerical columns.

```
# Summary statistics
print(customers.describe())
print(products.describe())
print(transactions.describe())

# Unique values in categorical columns
print(customers['Region'].value_counts())
```

```
print(products['Category'].value_counts())
```

Step 5: Handle Missing Values

Decide how to deal with missing data.

```
# Remove rows with missing values
transactions_cleaned = transactions.dropna()

# Fill missing values
customers['Region'] = customers['Region'].fillna('Unknown')
transactions['TotalValue'] = transactions['TotalValue'].fillna(transactions['TotalValue'].mean())
```

Step 6: Data Visualization

Use visualizations to understand patterns and relationships.

Univariate Analysis:

```
# Histogram for numerical data
transactions['TotalValue'].hist(bins=20)
plt.title('Distribution of Total Transaction Value')
plt.xlabel('Total Value')
plt.ylabel('Frequency')
plt.show()
```

```
# Bar chart for categorical data
customers['Region'].value_counts().plot(kind='bar')
plt.title('Customer Distribution by Region')
plt.xlabel('Region')
plt.ylabel('Count')
plt.show()
```

Bivariate Analysis:

```
# Scatter plot
sns.scatterplot(x='Price', y='TotalValue', data=transactions)
plt.title('Price vs Total Transaction Value')
plt.show()

# Boxplot
sns.boxplot(x='Category', y='Price', data=products)
plt.title('Price Distribution by Category')
plt.show()
```

Correlation Analysis:

```
# Correlation heatmap
correlation_matrix = transactions.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

Step 7: Identify Outliers

Detect and handle outliers in numerical data.

```
# Boxplot for outlier detection
sns.boxplot(transactions['TotalValue'])
plt.title('Boxplot of Total Transaction Value')
plt.show()

# Remove outliers using IQR
Q1 = transactions['TotalValue'].quantile(0.25)
Q3 = transactions['TotalValue'].quantile(0.75)
IQR = Q3 - Q1
transactions_cleaned = transactions[(transactions['TotalValue'] >= Q1 - 1.5 * IQR) &
                                    (transactions['TotalValue'] <= Q3 + 1.5 * IQR)]
```

Step 8: Merge Datasets

Combine datasets for deeper insights.

```
# Merge transactions with customers and products
merged_data = transactions.merge(customers, on='CustomerID').merge(products, on='ProductID')

# Preview merged data
print(merged_data.head())
```

Step 9: Advanced Analysis

Perform more detailed analyses based on business needs.

```
# Top 5 products by sales
top_products =
merged_data.groupby('ProductName')['TotalValue'].sum().sort_values(ascending=False).head(5)
print("Top 5 Products by Sales:\n", top_products)

# Sales trends over time
merged_data['TransactionDate'] = pd.to_datetime(merged_data['TransactionDate'])
sales_trends = merged_data.groupby(merged_data['TransactionDate'].dt.month)['TotalValue'].sum()
sales_trends.plot(kind='line')
plt.title('Monthly Sales Trends')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.show()
```

Step 10: Save Cleaned Data

Save the processed data for further use.

```
# Save cleaned data
merged_data.to_csv('cleaned_merged_data.csv', index=False)
```

Output Example

From EDA, you might derive insights like:

1. The top 5 products contribute 50% of total sales.

2. Customers from Asia account for the highest revenue.

3. Sales peak during certain months, indicating seasonal trends.

4. High-priced products have lower purchase frequency but higher revenue contribution.

5. Outliers in transaction values were removed, improving data quality.