```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
import numpy as np
import os
import logging

# Configure logging
logging.basicConfig(filename='shopping_analysis.log', level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# Load the dataset
data = pd.read_csv('shopping_data.csv')  # Replace with your actual dataset file name

# Log basic dataset information
logging.info("Dataset loaded successfully.")
logging.info(f"Number of rows: {data.shape[0]}, Number of columns: {data.shape[1]}")
logging.info(f"Columns: {list(data.columns)}")

# Data Cleaning
data.dropna(inplace=True)  # Remove rows with missing values
data['PurchaseDate'] = pd.to_datetime(data['PurchaseDate'])  # Convert to datetime

# Add new columns for analysis
data['Month'] = data['PurchaseDate'].dt.month
data['Year'] = data['PurchaseDate'].dt.year

# Analyze Monthly Sales Trends
monthly_sales = data.groupby(['Year', 'Month'])['Sales'].sum().reset_index()
plt.figure(figsize=(12, 6))
sns.lineplot(x='Month', y='Sales', hue='Year', data=monthly_sales)
plt.title('Monthly Sales Trends')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.savefig('monthly_sales_trends.png')
plt.show()

# Analyze Product Category Preferences
category_sales = data.groupby('Category')['Sales'].sum().sort_values(ascending=False)
plt.figure(figsize=(10, 5))
sns.barplot(x=category_sales.index, y=category_sales.values)
```

```
plt.title('Top-Selling Product Categories')
plt.xlabel('Category')
plt.ylabel('Sales')
plt.xticks(rotation=45)
plt.savefig('top_selling_categories.png')
plt.show()

# Cluster Customers Based on Spending
customer_data = data.groupby('CustomerID')['Sales'].sum().reset_index()

# Normalize Sales Data
scaler = StandardScaler()
customer_data['Sales'] = scaler.fit_transform(customer_data[['Sales']])

# Determine optimal number of clusters using the Elbow Method
inertia = []
silhouette_scores = []
k_range = range(2, 6)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(customer_data[['Sales']])
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(customer_data[['Sales']], labels))

optimal_k = k_range[np.argmax(silhouette_scores)]
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
customer_data['Cluster'] = kmeans.fit_predict(customer_data[['Sales']])

# Visualize Customer Clusters
plt.figure(figsize=(8, 5))
sns.scatterplot(x='CustomerID', y='Sales', hue='Cluster', data=customer_data, palette='viridis')
plt.title('Customer Segments')
plt.xlabel('Customer ID')
plt.ylabel('Total Spending (Normalized)')
plt.savefig('customer_segments.png')
plt.show()

# Save cleaned data
file_name = 'cleaned_shopping_data.csv'
if os.path.exists(file_name):
    logging.warning(f"{file_name} already exists. Consider renaming or backing up.")
    print(f"Warning: {file_name} already exists. Consider renaming or backing up.")
else:
```

```
data.to_csv(file_name, index=False)
logging.info(f"Data processing complete. Cleaned data saved as '{file_name}'.")
print(f"Data processing complete. Cleaned data saved as '{file_name}'.")
```