Python Script With Clustering Code

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
transactions_file = 'Transactions.xlsx'
products_file = 'Products.csv'

# Read the files
transactions_df = pd.read_excel(transactions_file, sheet_name='Transactions')
products_df = pd.read_csv(products_file)

# Merge transactions with product data
merged_data = pd.merge(transactions_df, products_df, on="ProductID", how="left")

# Create customer profiles
customer_profiles = merged_data.groupby("CustomerID").agg({
    "TransactionID": "count",  # Number of transactions
    "TotalValue": "sum",       # Total spending
    "Quantity": "sum",         # Total quantity purchased
    "Category": lambda x: x.mode()[0] if not x.mode().empty else None,  # Most frequent category
    "Price": "mean",           # Average price paid
}).rename(columns={
    "TransactionID": "NumTransactions",
    "TotalValue": "TotalSpent",
    "Quantity": "TotalQuantity",
```

```
    "Category": "PreferredCategory",
    "Price": "AvgPrice",
}).reset_index()

# One-hot encode the 'PreferredCategory' column
customer_profiles = pd.get_dummies(customer_profiles, columns=["PreferredCategory"],
drop_first=True)

# Prepare data for clustering
features = customer_profiles.drop(columns=["CustomerID"])
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Perform clustering
db_scores = []
clusters_range = range(2, 11)

for n_clusters in clusters_range:
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    labels = kmeans.fit_predict(scaled_features)
    db_index = davies_bouldin_score(scaled_features, labels)
    db_scores.append(db_index)

# Select the best number of clusters (lowest DB Index)
optimal_clusters = clusters_range[db_scores.index(min(db_scores))]
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
customer_profiles['Cluster'] = kmeans.fit_predict(scaled_features)

# Plot the DB Index for each number of clusters
plt.figure(figsize=(8, 5))
plt.plot(clusters_range, db_scores, marker='o', linestyle='--', color='b')
```

```
plt.title('Davies-Bouldin Index for Different Cluster Numbers')
plt.xlabel('Number of Clusters')
plt.ylabel('DB Index')
plt.xticks(clusters_range)
plt.grid()
plt.show()

# Visualize clusters using a pairplot
sns.pairplot(customer_profiles, hue='Cluster', palette='tab10')
plt.show()

# Print results
print(f"Optimal number of clusters: {optimal_clusters}")
print(f"Minimum DB Index: {min(db_scores)}")
print("Cluster-wise customer distribution:")
print(customer_profiles['Cluster'].value_counts())
```