```
import pandas as pd
import numpy as np
```

```
customers=pd.read_csv("/content/Customers.csv")
products=pd.read_csv("/content/Products.csv")
transactions=pd.read_csv("/content/Transactions.csv")
```

```
customer_transactions=pd.merge(customers,transactions,on="CustomerID",how="inner")
data=pd.merge(customer_transactions,products,on="ProductID",how="inner")
```

```
data.head()
```

	CustomerID	CustomerName	Region	SignupDate	TransactionID	ProductID	TransactionDate	Quantity	TotalValue	Price_x	ProductNa
0	C0001	Lawrence Carroll	South America	2022-07-10	T00015	P054	2024-01-19 03:12:55	2	114.60	57.30	SoundWa Cookbc
1	C0001	Lawrence Carroll	South America	2022-07-10	T00932	P022	2024-09-17 09:01:18	3	412.62	137.54	HomeSer Wall
2	C0001	Lawrence Carroll	South America	2022-07-10	T00085	P096	2024-04-08 00:01:00	2	614.94	307.47	SoundWa Headphor
3	C0001	Lawrence Carroll	South America	2022-07-10	T00445	P083	2024-05-07 03:11:44	2	911.44	455.72	ActiveW Smartwa
4	C0001	Lawrence Carroll	South America	2022-07-10	T00436	P029	2024-11-02 17:04:16	3	1300.92	433.64	TechF Headphor

Next steps: Generate code with data View recommended plots New interactive sheet

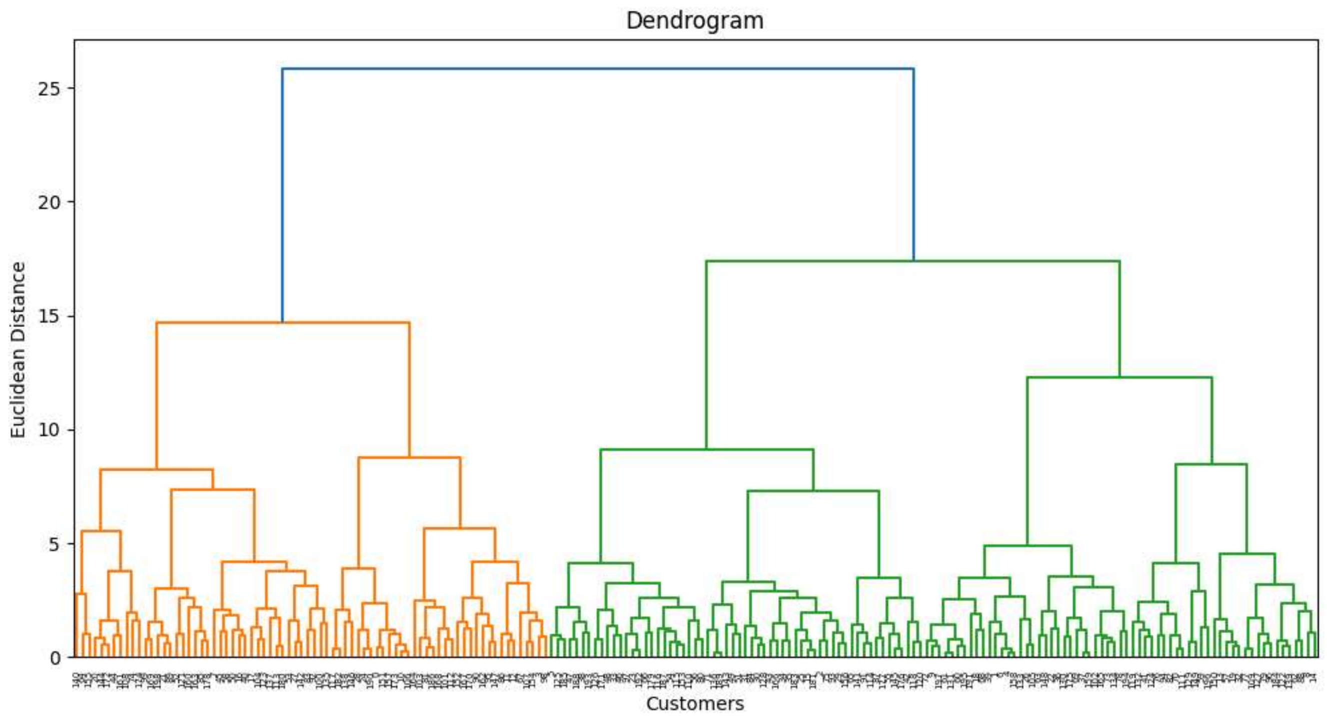
```
customer_data = data.groupby('CustomerID').agg({
    'TotalValue': 'sum',
    'Quantity': 'sum',
    'ProductID': 'nunique',
    'Category': 'nunique',
    'Region': 'first',
    'SignupDate': 'first'
}).reset_index()
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import davies_bouldin_score
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
import seaborn as sns
```

```
le = LabelEncoder()
customer_data['Region'] = le.fit_transform(customer_data['Region'])
customer_data['SignupDate'] = pd.to_datetime(customer_data['SignupDate'])
customer_data['DaysSinceSignup'] = (pd.Timestamp.now() - customer_data['SignupDate']).dt.days
customer_data.drop(columns=['SignupDate'], inplace=True)
```

```
features = ['TotalValue', 'Quantity', 'ProductID', 'Category', 'Region', 'DaysSinceSignup']
scaler = StandardScaler()
scaled_features = scaler.fit_transform(customer_data[features])
```

```
# Dendrogram to Determine Optimal Number of Clusters
plt.figure(figsize=(12, 6))
dendrogram = sch.dendrogram(sch.linkage(scaled_features, method='ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean Distance')
plt.show()
```

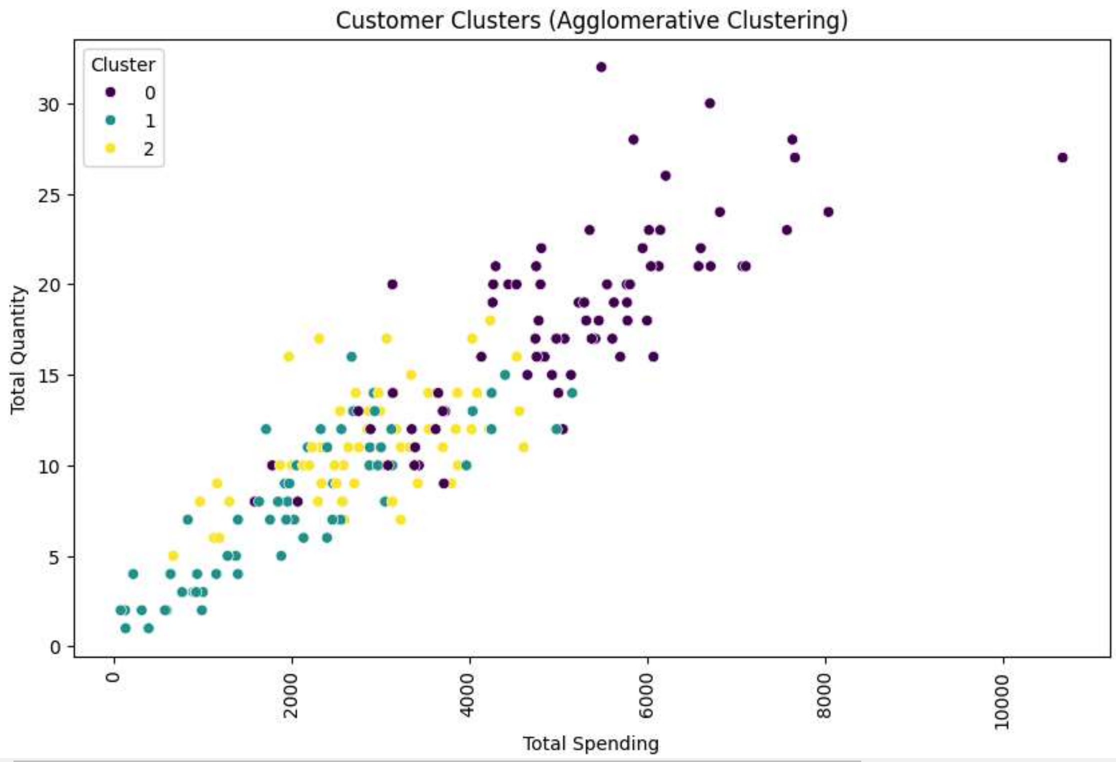


```
n_clusters = 3
hc = AgglomerativeClustering(n_clusters=n_clusters, linkage='ward')
customer_data['Cluster'] = hc.fit_predict(scaled_features)


db_index = davies_bouldin_score(scaled_features, customer_data['Cluster'])
print(f"Davies-Bouldin Index: {db_index}")
```

```
Davies-Bouldin Index: 1.559970755214448
```

```
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x=customer_data['TotalValue'],
    y=customer_data['Quantity'],
    hue=customer_data['Cluster'],
    palette='viridis'
)
plt.title("Customer Clusters (Agglomerative Clustering)")
plt.xlabel("Total Spending")
plt.ylabel("Total Quantity")
plt.xticks(rotation=90)
plt.legend(title="Cluster")
plt.show()
```



Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.