Clustering Logic and Performance Metrics

Clustering Logic Implementation

```
import os

# Set the environment variable to avoid memory leak warning
os.environ["OMP_NUM_THREADS"] = "1"

import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the datasets
customers = pd.read_csv('C:/Users/shara/OneDrive/Desktop/Zeotap/Customers.csv')
transactions = pd.read_csv('C:/Users/shara/OneDrive/Desktop/Zeotap/Transactions.csv')

# Merge customers and transactions
customer_transactions = pd.merge(transactions, customers, on='CustomerID', how='left')

# Create a feature for the total amount spent by each customer
customer_spending =
customer_transactions.groupby('CustomerID')['TotalValue'].sum().reset_index()
customer_spending.columns = ['CustomerID', 'TotalSpent']

# Merge the spending data back with the customers data
customers = pd.merge(customers, customer_spending, on='CustomerID',
how='left').fillna(0)

# Display the merged dataset
print(customers.head())
```

```
# Select features for clustering
features = ['TotalSpent']  # Add more features as needed

# Standardize the features
scaler = StandardScaler()
customer_data_scaled = scaler.fit_transform(customers[features])

# Determine the optimal number of clusters using the elbow method
wcss = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(customer_data_scaled)
    wcss.append(kmeans.inertia_)

# Plot the elbow curve
plt.plot(range(2, 11), wcss)
plt.title('Elbow Method For Optimal Clusters')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Performance Metrics

```
from sklearn.metrics import davies_bouldin_score

# Train the KMeans model with the optimal number of clusters
optimal_clusters = 4  # Choose based on the elbow method
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42)
customers['Cluster'] = kmeans.fit_predict(customer_data_scaled)

# Calculate the DB Index
db_index = davies_bouldin_score(customer_data_scaled, customers['Cluster'])
print('Davies-Bouldin Index:', db_index)
```

```
# Display clustering results

print(customers.head())
```

Visual representation of clusters.