Code Segmentation:

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

# Load the datasets
customers_df = pd.read_csv('Customers.csv')
transactions_df = pd.read_csv('Transactions.csv')

# Merge the datasets on CustomerID
merged_df = pd.merge(transactions_df, customers_df, on="CustomerID", how="inner")

# Convert TransactionDate and SignupDate to datetime format
merged_df['TransactionDate'] = pd.to_datetime(merged_df['TransactionDate'])
merged_df['SignupDate'] = pd.to_datetime(merged_df['SignupDate'])

# Define the current date as the max TransactionDate
current_date = merged_df['TransactionDate'].max()

# Create RFM features
rfm_df = merged_df.groupby('CustomerID').agg({
'TransactionDate': lambda x: (current_date - x.max()).days,  # Recency
'TransactionID': 'count',  # Frequency
'TotalValue': 'sum'  # Monetary Value
}).rename(columns={
'TransactionDate': 'Recency',
'TransactionID': 'Frequency',
'TotalValue': 'MonetaryValue'
}).reset_index()
```

```
# Scale the RFM features
scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm_df[['Recency', 'Frequency', 'MonetaryValue']])

# Determine the optimal number of clusters using Elbow Method and Silhouette Score
cluster_range = range(2, 11)
inertia = []
silhouette_scores = []

for k in cluster_range:
kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
kmeans.fit(rfm_scaled)
inertia.append(kmeans.inertia_)
silhouette_scores.append(silhouette_score(rfm_scaled, kmeans.labels_))

# Plot Elbow Method and Silhouette Scores
plt.figure(figsize=(14, 6))

# Elbow Method
plt.subplot(1, 2, 1)
plt.plot(cluster_range, inertia, marker='o')
plt.title('Elbow Method', fontsize=14)
plt.xlabel('Number of Clusters', fontsize=12)
plt.ylabel('Inertia', fontsize=12)
plt.grid(True)

# Silhouette Scores
plt.subplot(1, 2, 2)
plt.plot(cluster_range, silhouette_scores, marker='o', color='orange')
plt.title('Silhouette Scores', fontsize=14)
plt.xlabel('Number of Clusters', fontsize=12)
```

```
plt.ylabel('Silhouette Score', fontsize=12)
plt.grid(True)

plt.tight_layout()
plt.show()

# Finalize clustering with the optimal number of clusters (e.g., 3)
optimal_clusters = 3
final_kmeans = KMeans(n_clusters=optimal_clusters, random_state=42, n_init=10)
rfm_df['Cluster'] = final_kmeans.fit_predict(rfm_scaled)

# Analyze the clustering results
cluster_analysis = rfm_df.groupby('Cluster').agg({
'Recency': ['mean', 'std'],
'Frequency': ['mean', 'std'],
'MonetaryValue': ['mean', 'std']
}).round(2)

# Flatten MultiIndex for cleaner display
cluster_analysis.columns = ['_'.join(col).strip() for col in cluster_analysis.columns.values]
cluster_analysis.reset_index(inplace=True)

# Display cluster analysis
print("Cluster Analysis:")
print(cluster_analysis)
```

Output:

Cluster	Recency Mean	Recency Std	Frequency Mean
0	58.98	44.92	4.09
1	54.07	42.24	7.25
2	243.17	70.12	1.94