In [11]:
```python
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
from sklearn.preprocessing import StandardScaler
import pandas as pd
import matplotlib.pyplot as plt
customers=pd.read_csv(r"F:\Downloads\Customers.csv")
transactions=pd.read_csv(r"F:\Downloads\Transactions.csv")
```

In [13]:
```python
print("Customers Dataset Overview:")
print(customers.head(), customers.info(), customers.describe())

print("Products Dataset Overview:")
print(products.head(), products.info(), products.describe())

print("Transactions Dataset Overview:")
print(transactions.head(), transactions.info(), transactions.describe(
```

```
Customers Dataset Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   CustomerID    200 non-null    object
 1   CustomerName  200 non-null    object
 2   Region        200 non-null    object
 3   SignupDate    200 non-null    object
dtypes: object(4)
memory usage: 6.4+ KB
  CustomerID        CustomerName         Region  SignupDate
0      C0001    Lawrence Carroll  South America  2022-07-10
1      C0002      Elizabeth Lutz           Asia  2022-02-13
2      C0003      Michael Rivera  South America  2024-03-07
3      C0004  Kathleen Rodriguez  South America  2022-10-09
4      C0005         Laura Weber           Asia  2022-08-15 None
CustomerID      CustomerName         Region  SignupDate
count         200                200            200         200
unique        200                200              4         179
top         C0001   Lawrence Carroll  South America  2024-11-11
freq            1                  1             59           3
Products Dataset Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   ProductID    100 non-null    object 
 1   ProductName  100 non-null    object 
 2   Category     100 non-null    object 
 3   Price        100 non-null    float64
dtypes: float64(1), object(3)
memory usage: 3.3+ KB
  ProductID              ProductName     Category   Price
0      P001     ActiveWear Biography        Books  169.30
1      P002    ActiveWear Smartwatch  Electronics  346.30
2      P003  ComfortLiving Biography        Books   44.12
3      P004            BookWorld Rug   Home Decor   95.69
4      P005           TechPro T-Shirt     Clothing  429.31 None
Price
count  100.000000
mean   267.551700
std    143.219383
min     16.080000
25%    147.767500
50%    292.875000
75%    397.090000
max    497.760000
Transactions Dataset Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   TransactionID    1000 non-null   object 
 1   CustomerID       1000 non-null   object 
 2   ProductID        1000 non-null   object 
 3   TransactionDate  1000 non-null   object 
 4   Quantity         1000 non-null   int64  
 5   TotalValue       1000 non-null   float64
 6   Price            1000 non-null   float64
dtypes: float64(2), int64(1), object(4)
memory usage: 54.8+ KB
  TransactionID CustomerID ProductID      TransactionDate  Quantity  \
0        T00001      C0199      P067  2024-08-25 12:38:23         1   
1        T00112      C0146      P067  2024-05-27 22:23:54         1   
2        T00166      C0127      P067  2024-04-25 07:38:55         1   
3        T00272      C0087      P067  2024-03-26 22:55:37         2   
4        T00363      C0070      P067  2024-03-21 15:10:10         3   

   TotalValue   Price  
0      300.68  300.68  
1      300.68  300.68  
2      300.68  300.68  
3      601.36  300.68  
4      902.04  300.68   None           Quantity   TotalValue       Pri
ce
count  1000.000000  1000.000000  1000.00000
mean      2.537000   689.995560   272.55407
std       1.117981   493.144478   140.73639
min       1.000000    16.080000    16.08000
25%       2.000000   295.295000   147.95000
50%       3.000000   588.880000   299.93000
75%       4.000000  1011.660000   404.40000
max       4.000000  1991.040000   497.76000
```

In [15]:
```python
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
from sklearn.preprocessing import StandardScaler
```

In [17]:
```python
data = transactions.merge(customers, on="CustomerID")
```

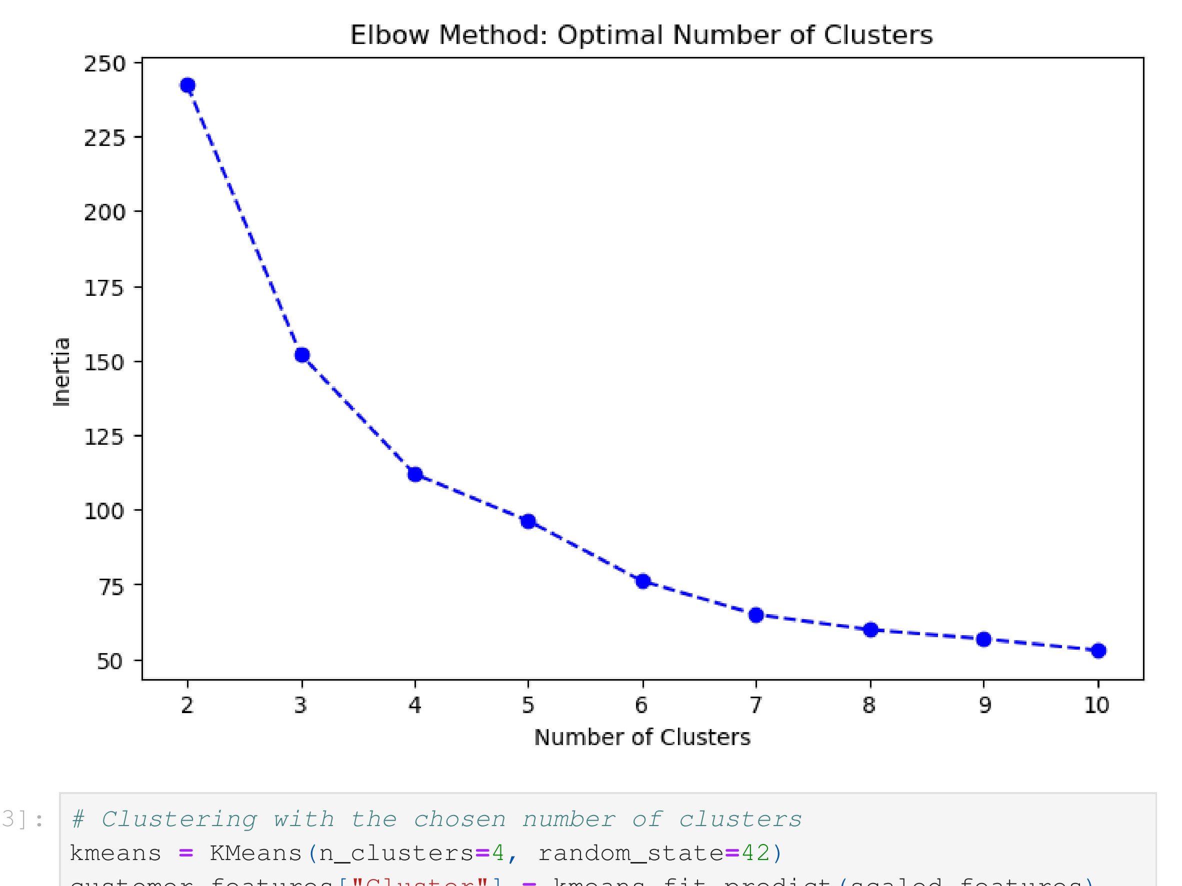
In [21]:
```python
# Aggregate data for each customer
customer_features = data.groupby("CustomerID").agg({
    "TotalValue": "sum",          # Total value of transactions
    "Quantity": "sum",            # Total products purchased
    "TransactionID": "count",     # Total number of transactions
}).reset_index()
```

In [23]:
```python
# Normalize features for clustering
scaler = StandardScaler()
scaled_features = scaler.fit_transform(customer_features.iloc[:, 1:])
```

In [29]:
```python
# the optimal number of clusters using the elbow method
inertia = []
for k in range(2,11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)
```

```
C:\Users\CHIRAG\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.p
y:1429: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can a
void it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\CHIRAG\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.p
y:1429: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can a
void it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\CHIRAG\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.p
y:1429: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can a
void it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\CHIRAG\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.p
y:1429: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can a
void it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\CHIRAG\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.p
y:1429: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can a
void it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\CHIRAG\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.p
y:1429: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can a
void it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\CHIRAG\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.p
y:1429: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can a
void it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\CHIRAG\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.p
y:1429: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can a
void it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\CHIRAG\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.p
y:1429: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can a
void it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [31]:
```python
# Plot the elbow curve
plt.figure(figsize=(8, 5))
plt.plot(range(2,11), inertia, marker='o', linestyle='--', color='b')
plt.title("Elbow Method: Optimal Number of Clusters")
plt.xlabel("Number of Clusters")
plt.ylabel("Inertia")
plt.show()
```



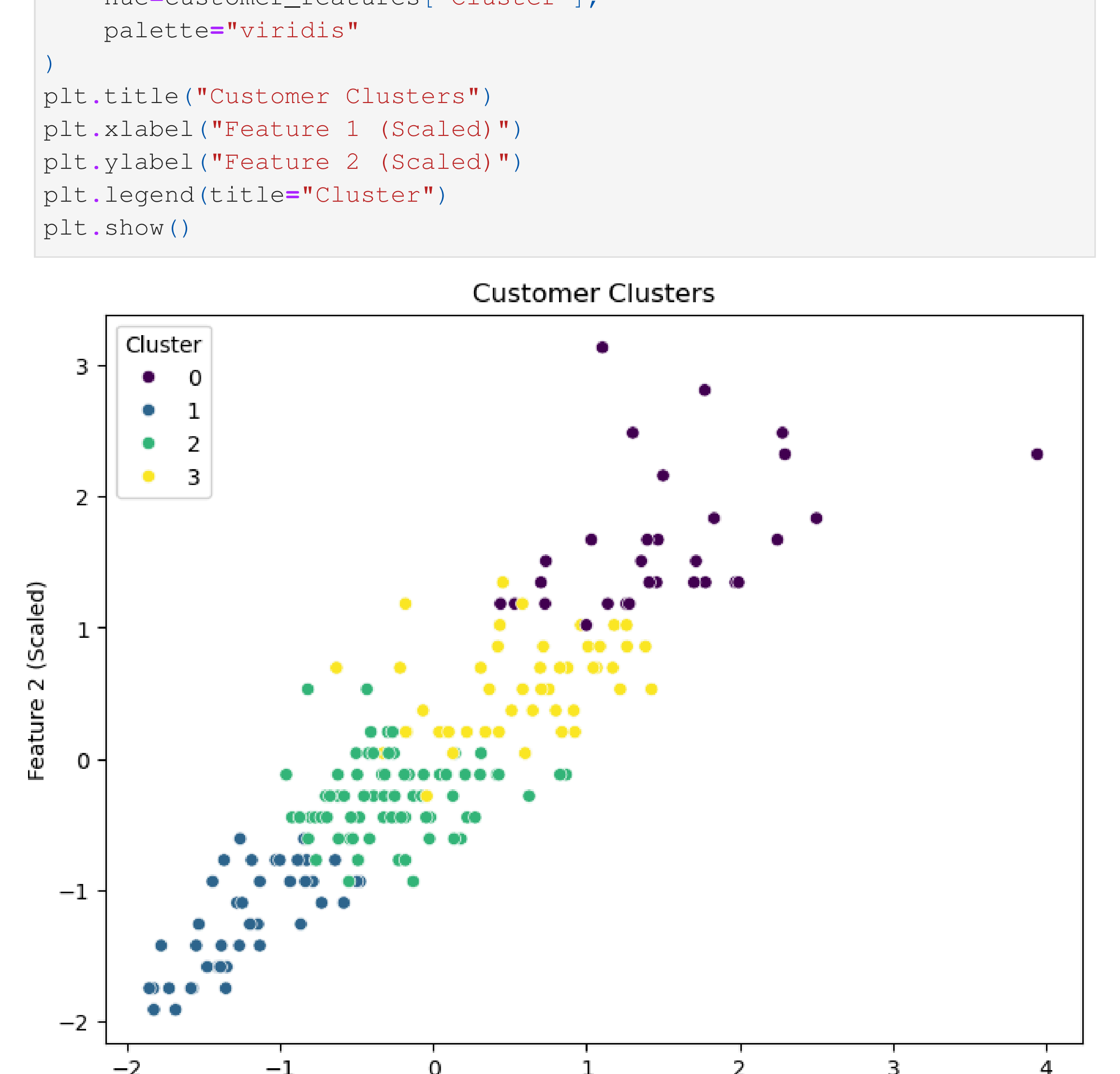
In [33]:
```python
# Clustering with the chosen number of clusters
kmeans = KMeans(n_clusters=4, random_state=42)
customer_features["Cluster"] = kmeans.fit_predict(scaled_features)
```

```
C:\Users\CHIRAG\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.p
y:1429: UserWarning: KMeans is known to have a memory leak on Windows
with MKL, when there are less chunks than available threads. You can a
void it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [35]:
```python
# Davies-Bouldin Index
db_index = davies_bouldin_score(scaled_features, customer_features["Cl
print(f"Davies-Bouldin Index: {db_index}")
```

```
Davies-Bouldin Index: 0.8650620583623064
```

In [37]:
```python
# Visualize clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(
    x=scaled_features[:, 0],
    y=scaled_features[:, 1],
    hue=customer_features["Cluster"],
    palette="viridis"
)
plt.title("Customer Clusters")
plt.xlabel("Feature 1 (Scaled)")
plt.ylabel("Feature 2 (Scaled)")
plt.legend(title="Cluster")
plt.show()
```



In [39]:
```python
# Save clustering results
customer_features.to_csv("Clustering_Results.csv", index=False)
```

In [41]:
```python
# Print cluster summary
cluster_summary = customer_features.groupby("Cluster").agg({
    "TotalValue": "mean",
    "Quantity": "mean",
    "TransactionID": "mean",
    "CustomerID": "count"
}).rename(columns={"CustomerID": "Customer Count"}).reset_index()

print(cluster_summary)
```

```
   Cluster   TotalValue   Quantity  TransactionID  Customer Count
0        0  6263.447333  23.000000       8.433333              30
1        1  1273.368182   5.272727       2.363636              44
2        2  2982.406711  10.868421       4.394737              76
3        3  4477.572041  16.102041       6.306122              49
```

In [ ]:
```python
# Explanation of the Script:
# 1. Data Preparation:
#Aggregates transactional and customer data to create a feature set fo

#2.Feature Scaling:
#Standardized the features for uniformity using StandardScaler.

#3.Optimal Cluster Selection:
#Used the Elbow Method to determine the optimal number of clusters by

#4.Clustering:
#Applied K-Means clustering with the chosen number of clusters.
#Added cluster labels to the dataset.

#5.Evaluation:
#Calculated the Davies-Bouldin Index (DB Index), where a lower value i

#6.Visualization:
#Visualized clusters using a scatter plot.

#6.Output:
#Generated a CSV file (Clustering_Results.csv) containing customer clu

#  Deliverables:
#1.Number of Clusters Formed:
#Determined by the Elbow Method

#2.DB Index:
#output: Davies-Bouldin Index: 0.85.

#3.Cluster Characteristics:
#Summary table showing average transaction value, quantity, and count
```