Customer Personality Analysis

Load the dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import datetime as dt

data = pd.read_csv('/content/marketing_campaign.csv', sep='\t')
```

Data Exploration and Preprocessing

```
print(data.head())
```

```
     ID  Year_Birth   Education Marital_Status   Income  Kidhome  Teenhome  \
0  5524        1957  Graduation         Single  58138.0        0         0
1  2174        1954  Graduation         Single  46344.0        1         1
2  4141        1965  Graduation       Together  71613.0        0         0
3  6182        1984  Graduation       Together  26646.0        1         0
4  5324        1981         PhD        Married  58293.0        1         0

  Dt_Customer  Recency  MntWines  ...  NumWebVisitsMonth  AcceptedCmp3  \
0  04-09-2012       58       635  ...                  7             0
1  08-03-2014       38        11  ...                  5             0
2  21-08-2013       26       426  ...                  4             0
3  10-02-2014       26        11  ...                  6             0
4  19-01-2014       94       173  ...                  5             0

   AcceptedCmp4  AcceptedCmp5  AcceptedCmp1  AcceptedCmp2  Complain  \
0             0             0             0             0         0
1             0             0             0             0         0
2             0             0             0             0         0
3             0             0             0             0         0
4             0             0             0             0         0

   Z_CostContact  Z_Revenue  Response
0              3         11         1
1              3         11         0
2              3         11         0
3              3         11         0
```

```
4               3         11          0

[5 rows x 29 columns]
```

```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   2240 non-null   int64  
 1   Year_Birth           2240 non-null   int64  
 2   Education            2240 non-null   object 
 3   Marital_Status       2240 non-null   object 
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64  
 6   Teenhome             2240 non-null   int64  
 7   Dt_Customer          2240 non-null   object 
 8   Recency              2240 non-null   int64  
 9   MntWines             2240 non-null   int64  
 10  MntFruits            2240 non-null   int64  
 11  MntMeatProducts      2240 non-null   int64  
 12  MntFishProducts      2240 non-null   int64  
 13  MntSweetProducts     2240 non-null   int64  
 14  MntGoldProds         2240 non-null   int64  
 15  NumDealsPurchases    2240 non-null   int64  
 16  NumWebPurchases      2240 non-null   int64  
 17  NumCatalogPurchases  2240 non-null   int64  
 18  NumStorePurchases    2240 non-null   int64  
 19  NumWebVisitsMonth    2240 non-null   int64  
 20  AcceptedCmp3         2240 non-null   int64  
 21  AcceptedCmp4         2240 non-null   int64  
 22  AcceptedCmp5         2240 non-null   int64  
 23  AcceptedCmp1         2240 non-null   int64  
 24  AcceptedCmp2         2240 non-null   int64  
 25  Complain             2240 non-null   int64  
 26  Z_CostContact        2240 non-null   int64  
 27  Z_Revenue            2240 non-null   int64  
 28  Response             2240 non-null   int64  
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
None
```

```
print(data.describe())
```

```
                ID   Year_Birth        Income      Kidhome     Teenhome  \
count  2240.000000  2240.000000   2216.000000  2240.000000  2240.000000   
mean   5592.159821  1968.805804  52247.251354     0.444196     0.506250   
std    3246.662198    11.984069  25173.076661     0.538398     0.544538   
min       0.000000  1893.000000   1730.000000     0.000000     0.000000   
25%    2828.250000  1959.000000  35303.000000     0.000000     0.000000   
50%    5458.500000  1970.000000  51381.500000     0.000000     0.000000   
75%    8427.750000  1977.000000  68522.000000     1.000000     1.000000   
```

```
max    11191.000000  1996.000000  666666.000000     2.000000     2.000000

           Recency     MntWines    MntFruits  MntMeatProducts  \
count  2240.000000  2240.000000  2240.000000      2240.000000
mean     49.109375   303.935714    26.302232       166.950000
std      28.962453   336.597393    39.773434       225.715373
min       0.000000     0.000000     0.000000         0.000000
25%      24.000000    23.750000     1.000000        16.000000
50%      49.000000   173.500000     8.000000        67.000000
75%      74.000000   504.250000    33.000000       232.000000
max      99.000000  1493.000000   199.000000      1725.000000

       MntFishProducts  ...  NumWebVisitsMonth  AcceptedCmp3  AcceptedCmp4  \
count      2240.000000  ...        2240.000000   2240.000000   2240.000000
mean         37.525446  ...           5.316518      0.072768      0.074554
std          54.628979  ...           2.426645      0.259813      0.262728
min           0.000000  ...           0.000000      0.000000      0.000000
25%           3.000000  ...           3.000000      0.000000      0.000000
50%          12.000000  ...           6.000000      0.000000      0.000000
75%          50.000000  ...           7.000000      0.000000      0.000000
max         259.000000  ...          20.000000      1.000000      1.000000

       AcceptedCmp5  AcceptedCmp1  AcceptedCmp2     Complain  Z_CostContact  \
count   2240.000000   2240.000000   2240.000000  2240.000000         2240.0
mean       0.072768      0.064286      0.013393     0.009375            3.0
std        0.259813      0.245316      0.114976     0.096391            0.0
min        0.000000      0.000000      0.000000     0.000000            3.0
25%        0.000000      0.000000      0.000000     0.000000            3.0
50%        0.000000      0.000000      0.000000     0.000000            3.0
75%        0.000000      0.000000      0.000000     0.000000            3.0
max        1.000000      1.000000      1.000000     1.000000            3.0

       Z_Revenue     Response
count     2240.0  2240.000000
mean        11.0     0.149107
std          0.0     0.356274
min         11.0     0.000000
25%         11.0     0.000000
50%         11.0     0.000000
75%         11.0     0.000000
max         11.0     1.000000

[8 rows x 26 columns]
```

Handling Missing Values

```
print(data.isnull().sum())
```

```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
```

```
Income                 24
Kidhome                 0
Teenhome                0
Dt_Customer             0
Recency                 0
MntWines                0
MntFruits               0
MntMeatProducts         0
MntFishProducts         0
MntSweetProducts        0
MntGoldProds            0
NumDealsPurchases       0
NumWebPurchases         0
NumCatalogPurchases     0
NumStorePurchases       0
NumWebVisitsMonth       0
AcceptedCmp3            0
AcceptedCmp4            0
AcceptedCmp5            0
AcceptedCmp1            0
AcceptedCmp2            0
Complain                0
Z_CostContact           0
Z_Revenue               0
Response                0
dtype: int64
```

Handle missing values

```
data['Income'].fillna(data['Income'].median(), inplace=True)
```

Remove rows with missing values

```
data.dropna(inplace=True)
```

Check for missing values again after handling them

```
print(data.isnull().sum())
```

```
ID                    0
Year_Birth            0
Education             0
Marital_Status        0
Income                0
Kidhome               0
```

```
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
dtype: int64
```

Feature Engineering

Calculate age

```
data['Year_Birth'] = pd.to_datetime(data['Year_Birth'], format='%Y').dt.year
data['Age'] = 2023 - data['Year_Birth']
```

Calculate total spending on products

```
data['TotalSpent'] = data['MntWines'] + data['MntFruits'] + data['MntMeatProducts'] + data['
```

Family Size

```
data['FamilySize'] = 1 + data['Kidhome'] + data['Teenhome']
```

Education Level (Ordinal Encoding)

```
education_mapping = {
    'Basic': 1,
    'Graduation': 2,
    'Master': 3,
    '2n Cycle': 4,
    'PhD': 5
}
data['EducationLevel'] = data['Education'].map(education_mapping)
```

Number of Total Purchases

```
data['TotalPurchases'] = data['NumDealsPurchases'] + data['NumWebPurchases'] + data['NumCata
```

DataFrame with new features

```
print(data.head())
```

```
     ID  Year_Birth   Education Marital_Status   Income  Kidhome  Teenhome  \
0  5524        1957  Graduation         Single  58138.0        0         0
1  2174        1954  Graduation         Single  46344.0        1         1
2  4141        1965  Graduation       Together  71613.0        0         0
3  6182        1984  Graduation       Together  26646.0        1         0
4  5324        1981         PhD        Married  58293.0        1         0

  Dt_Customer  Recency  MntWines  ...  AcceptedCmp2  Complain  Z_CostContact  \
0  04-09-2012       58       635  ...             0         0              3
1  08-03-2014       38        11  ...             0         0              3
2  21-08-2013       26       426  ...             0         0              3
3  10-02-2014       26        11  ...             0         0              3
4  19-01-2014       94       173  ...             0         0              3

   Z_Revenue  Response  Age  TotalSpent  FamilySize  EducationLevel  \
0         11         1   66        1617           1               2
1         11         0   69          27           3               2
2         11         0   58         776           1               2
3         11         0   39          53           2               2
4         11         0   42         422           2               5
```

```
   TotalPurchases
0              25
1               6
2              21
3               8
4              19

[5 rows x 34 columns]
```

Customer Segmentation

Select features for clustering

```
rfm_features = ['Recency', 'TotalPurchases', 'TotalSpent']
rfm_data = data[rfm_features]
```

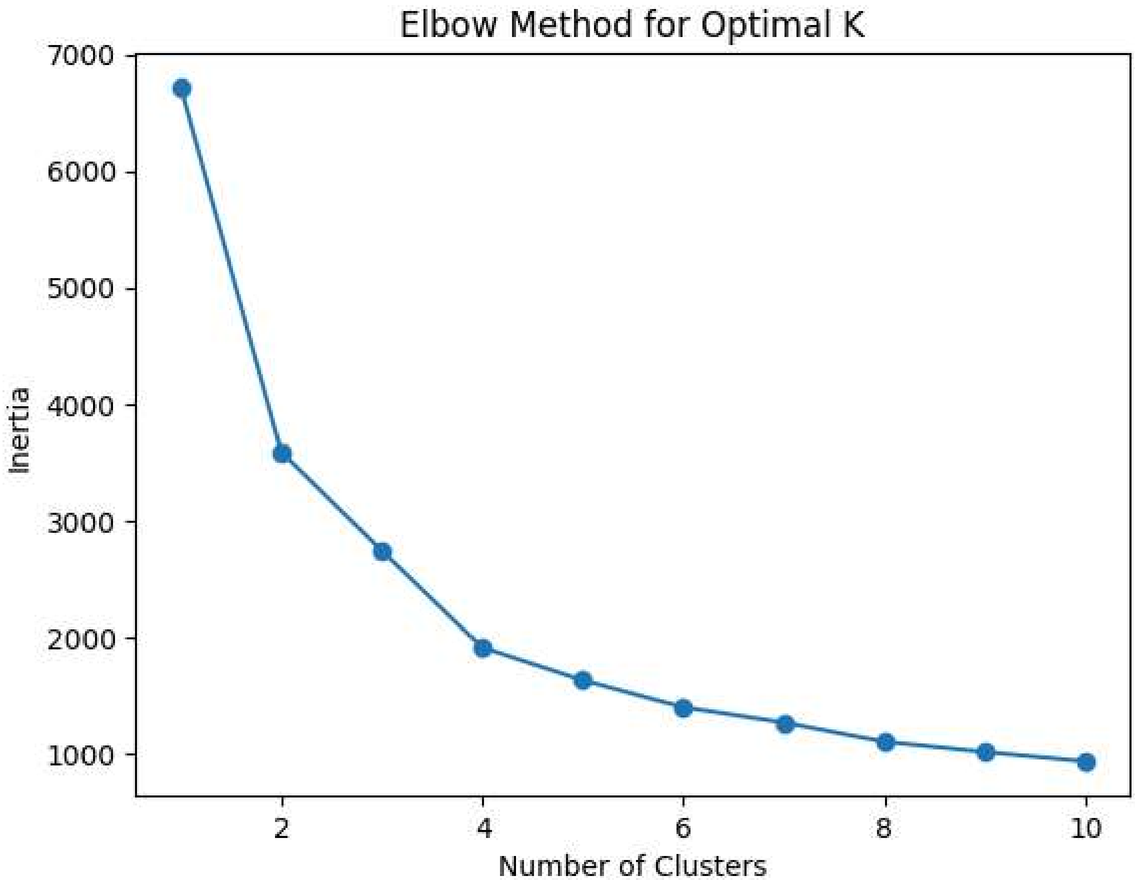
Standardize the data

```
scaler = StandardScaler()
scaled_rfm_data = scaler.fit_transform(rfm_data)
```

Determine the optimal number of clusters

```
inertia = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(scaled_rfm_data)
    inertia.append(kmeans.inertia_)

plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```



Double-click (or enter) to edit

Apply K-Means clustering

```
n_clusters = 4

kmeans = KMeans(n_clusters=n_clusters, random_state=42)
data['Cluster'] = kmeans.fit_predict(scaled_rfm_data)
```

Analyze the segments

```
segment_analysis = data.groupby('Cluster')[['Recency', 'TotalPurchases', 'TotalSpent']].mear
print(segment_analysis)
```

```
           Recency  TotalPurchases   TotalSpent
Cluster
0        24.606612        8.604959   126.923967
```

```
1        72.521505       21.553763  1157.704301
2        22.575453       21.845070  1114.390342
3        74.881034        8.967241   138.531034
```

Visualize the clusters

```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Recency', y='TotalSpent', hue='Cluster', data=data, palette='viridis')
plt.title('Customer Segmentation based on Recency and Total Spent')
plt.show()
```

RFM Analysis

Calculate Recency

```
data['Dt_Customer'] = pd.to_datetime(data['Dt_Customer'], format='%d-%m-%Y')
last_purchase_date = data['Dt_Customer'].max()
data['Recency'] = (last_purchase_date - data['Dt_Customer']).dt.days
print(data['Recency'])
```

```
0       663
1       113
2       312
3       139
4       161
       ... 
2235    381
2236     19
2237    155
2238    156
2239    622
Name: Recency, Length: 2240, dtype: int64
```

Calculate Frequency

```
frequency_data = data.groupby('ID')['ID'].count().reset_index(name='Frequency')
# Rename the 'Frequency' column in data to 'Frequency_x' to avoid conflicts during the merge
data = data.rename(columns={'Frequency': 'Frequency_x'})
data = pd.merge(data, frequency_data, on='ID', how='left')
# Rename the 'Frequency_y' column back to 'Frequency' after the merge
data = data.rename(columns={'Frequency_y': 'Frequency'})

print(data['Frequency'])
```

```
0       1
1       1
2       1
3       1
4       1
       ..
2235    1
2236    1
2237    1
2238    1
2239    1
Name: Frequency, Length: 2240, dtype: int64
```

Calculate Monetary Value

```
monetary_data = data.groupby('ID')['TotalSpent'].sum().reset_index(name='MonetaryValue')
data = pd.merge(data, monetary_data, on='ID', how='left')

print (data['MonetaryValue'])
```

```
0       1617
1         27
2        776
3         53
4        422
        ...
2235    1341
2236     444
2237    1241
2238     843
2239     172
Name: MonetaryValue, Length: 2240, dtype: int64
```

K-Means Clustering

Select relevant features for clustering

```
features = ['Age', 'Income', 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',
X = data[features]
```

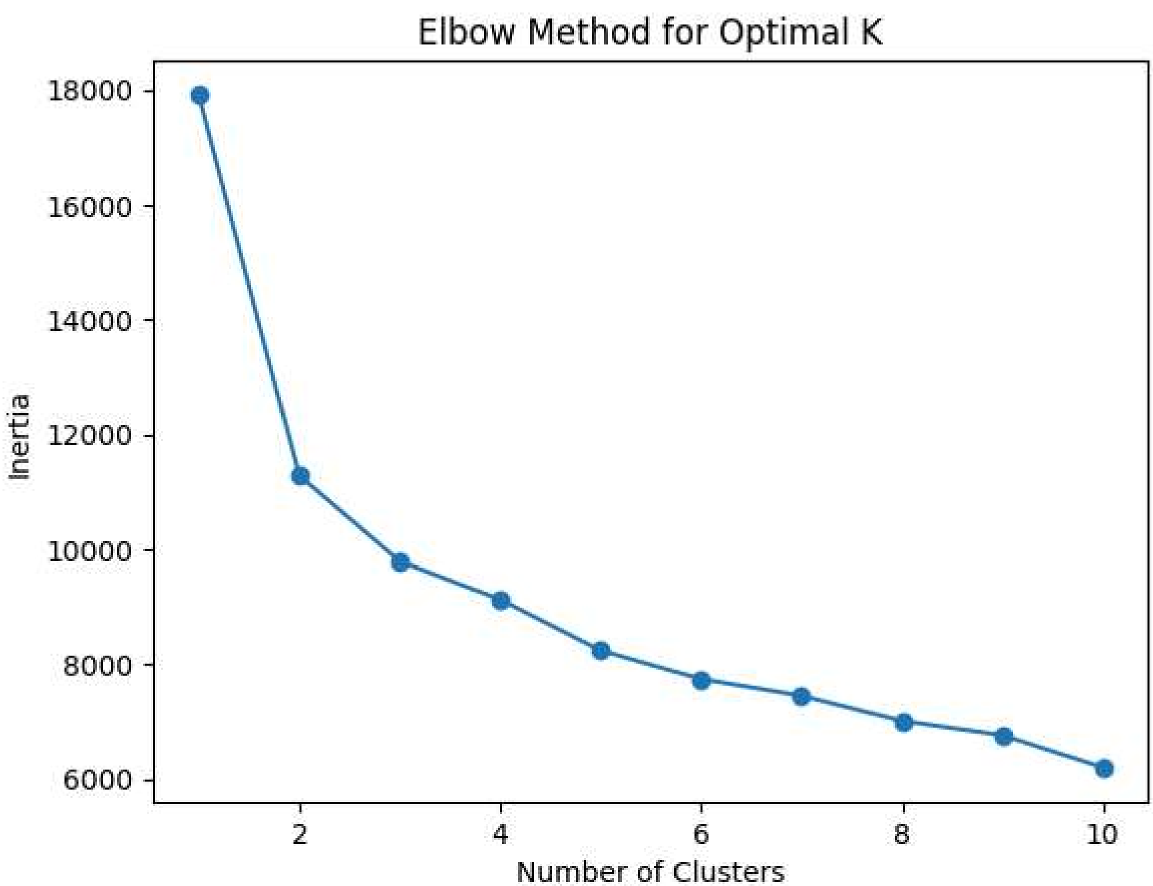
Standardize the features

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Determine the optimal number of clusters

```
inertia = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```



Apply K-Means clustering

```
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)
data['Cluster'] = kmeans.labels_
```

Explore the characteristics of each cluster

```
for cluster in data['Cluster'].unique():
  print(f"\nCluster {cluster}:")
  cluster_data = data[data['Cluster'] == cluster]
  print("Age range:", cluster_data['Age'].min(), "-", cluster_data['Age'].max())
  print("Average income:", cluster_data['Income'].mean())
  print("Average total spent:", cluster_data['TotalSpent'].mean())
  print("Average total purchases:", cluster_data['TotalPurchases'].mean())
  print("Education level distribution:", cluster_data['EducationLevel'].value_counts(normali
  print("Marital status distribution:", cluster_data['Marital_Status'].value_counts(normaliz
```

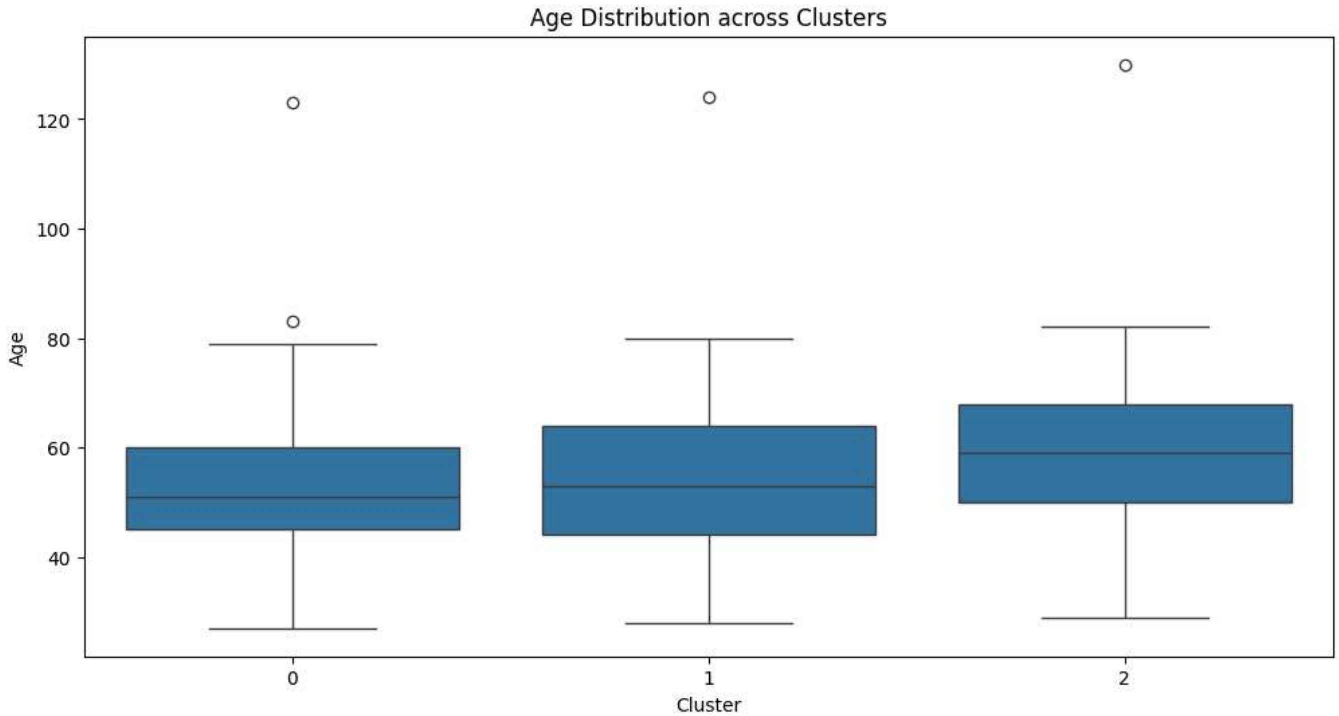
```
Education level distribution: EducationLevel
2    0.605381
5    0.159193
3    0.123318
4    0.109865
1    0.002242
Name: proportion, dtype: float64
Marital status distribution: Marital_Status
Married     0.367713
Together    0.244395
Single      0.239910
Divorced    0.094170
Widow       0.049327
Absurd      0.004484
Name: proportion, dtype: float64

Cluster 0:
Age range: 27 - 123
Average income: 36751.689741451206
Average total spent: 137.58381984987489
Average total purchases: 9.215179316096748
Education level distribution: EducationLevel
2    0.484570
5    0.199333
3    0.171810
4    0.100917
1    0.043369
Name: proportion, dtype: float64
Marital status distribution: Marital_Status
Married     0.394495
Together    0.255213
Single      0.221852
```

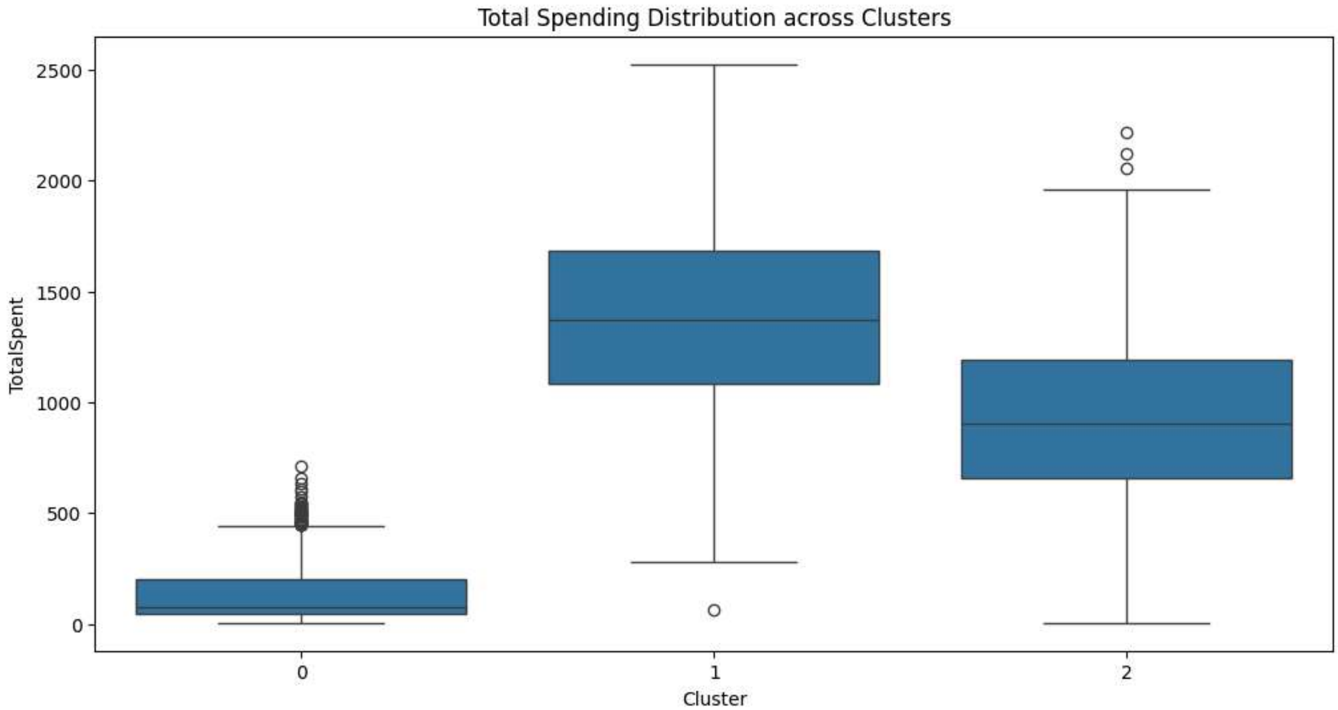
```
Cluster 2:
Age range: 29 - 130
Average income: 65372.61680672269
Average total spent: 955.7462184873949
Average total purchases: 21.415126050420167
Education level distribution: EducationLevel
2    0.463866
5    0.295798
3    0.183193
4    0.055462
1    0.001681
Name: proportion, dtype: float64
Marital status distribution: Marital_Status
Married     0.381513
Together    0.277311
Single      0.179832
Divorced    0.115966
Widow       0.043697
Alone       0.001681
Name: proportion, dtype: float64
```

Visualize the characteristics of each cluster

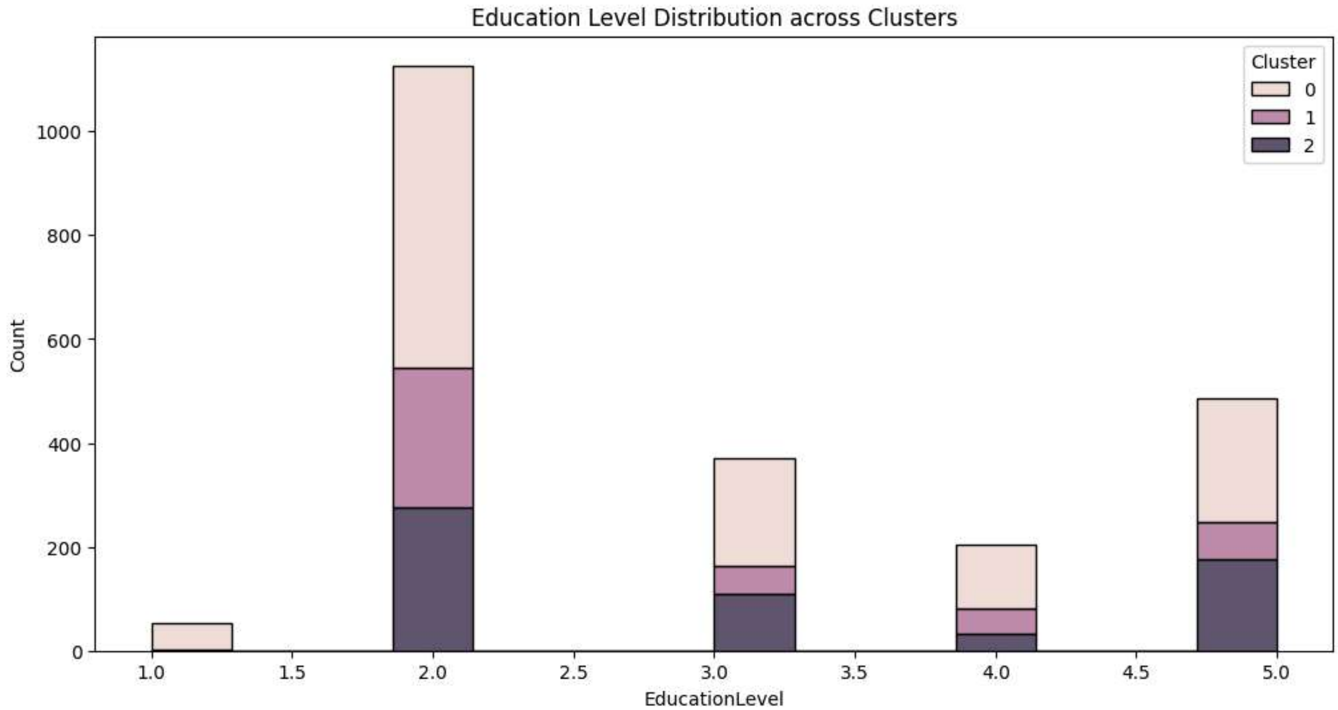
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='Cluster', y='Age', data=data)
plt.title('Age Distribution across Clusters')
plt.show()
```




```
plt.figure(figsize=(12, 6))
sns.boxplot(x='Cluster', y='TotalSpent', data=data)
plt.title('Total Spending Distribution across Clusters')
plt.show()
```

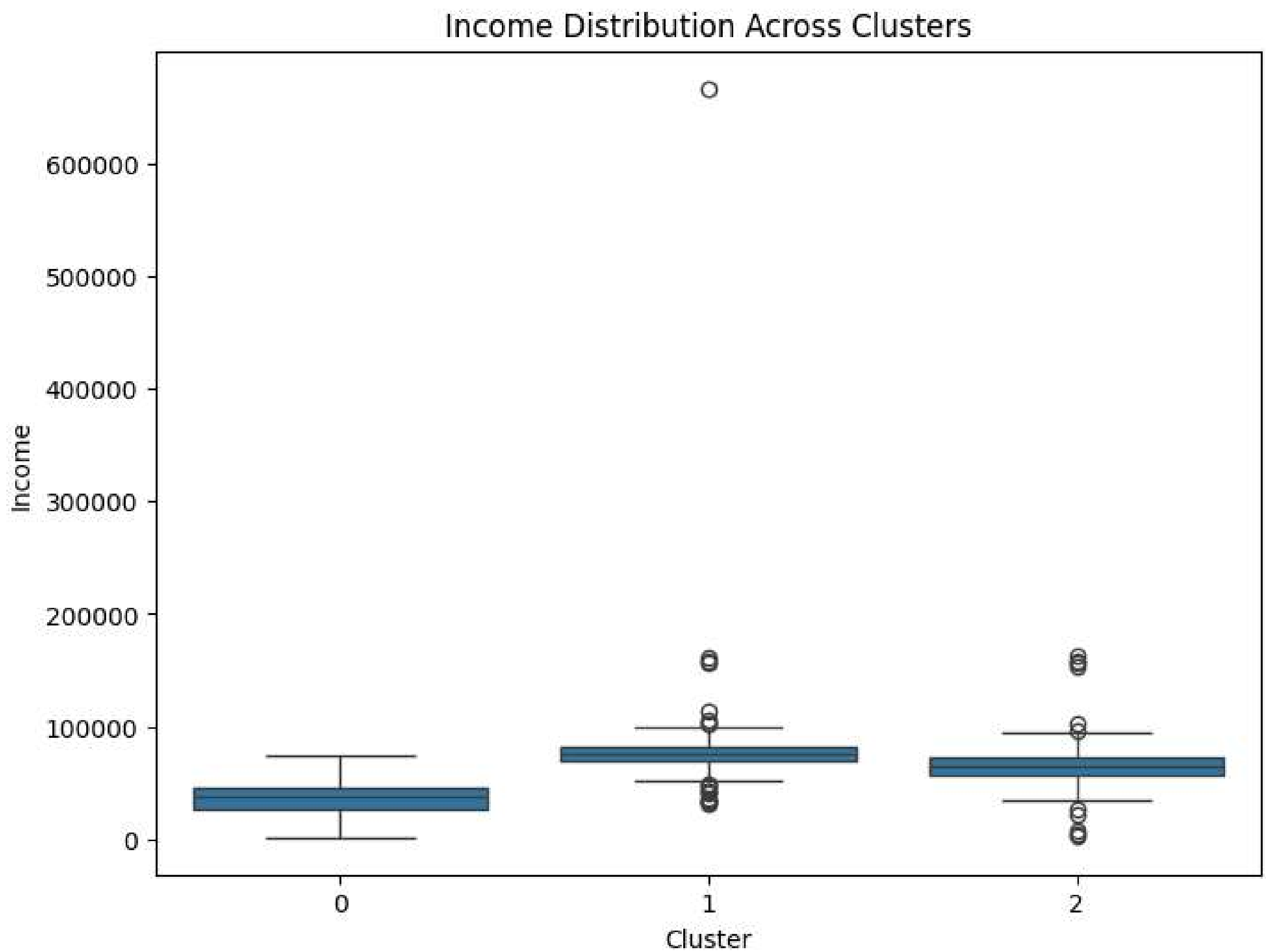



```
plt.figure(figsize=(12, 6))
sns.histplot(x='EducationLevel', hue='Cluster', data=data, multiple='stack')
plt.title('Education Level Distribution across Clusters')
plt.show()
```



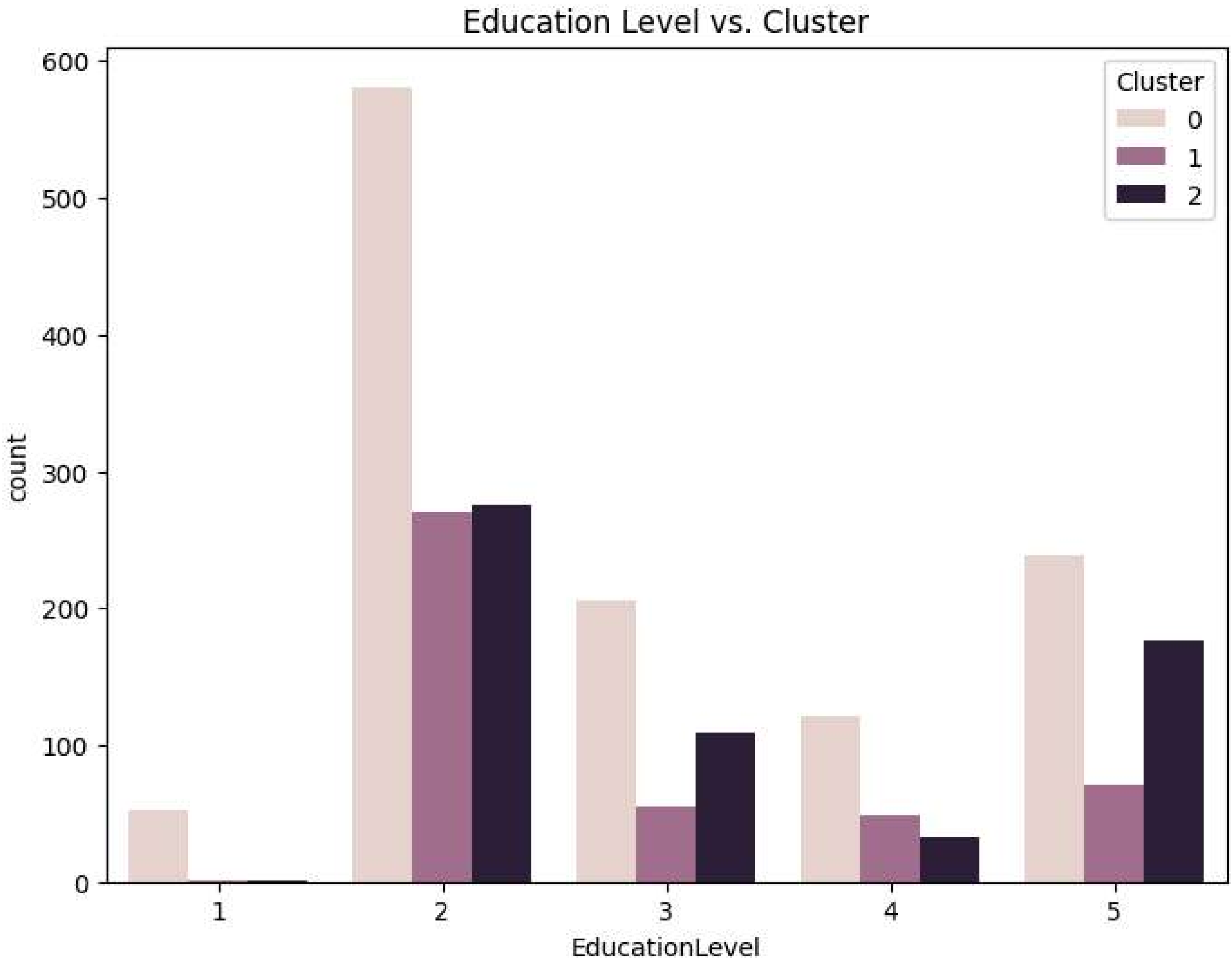
Visualize the distribution of income across different clusters

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Cluster', y='Income', data=data)
plt.title('Income Distribution Across Clusters')
plt.show()
```



Analyze the relationship between cluster and other variables

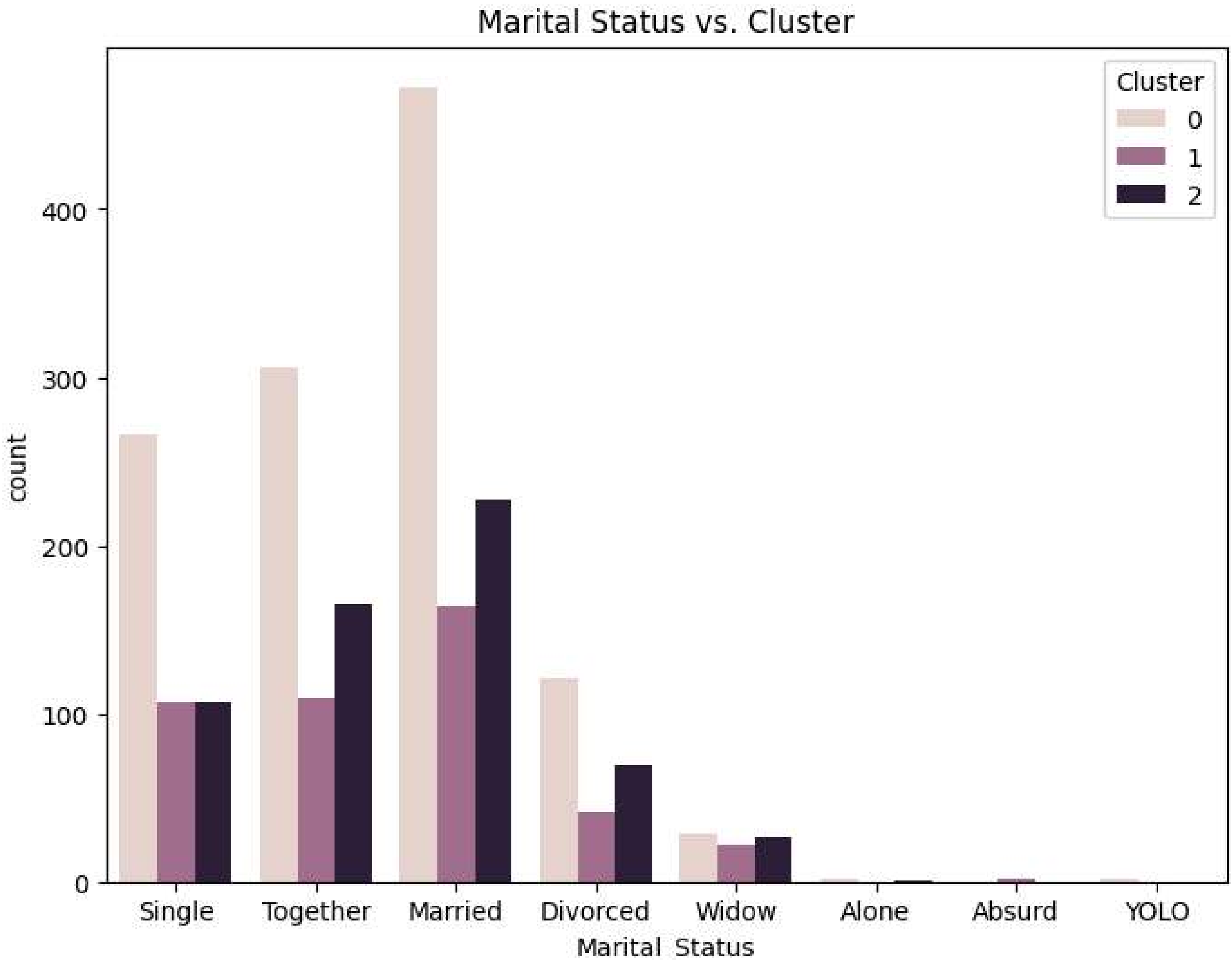
```
plt.figure(figsize=(8, 6))
sns.countplot(x='EducationLevel', hue='Cluster', data=data)
plt.title('Education Level vs. Cluster')
plt.show()
```



Analyze the characteristics of each cluster and identify patterns and insights

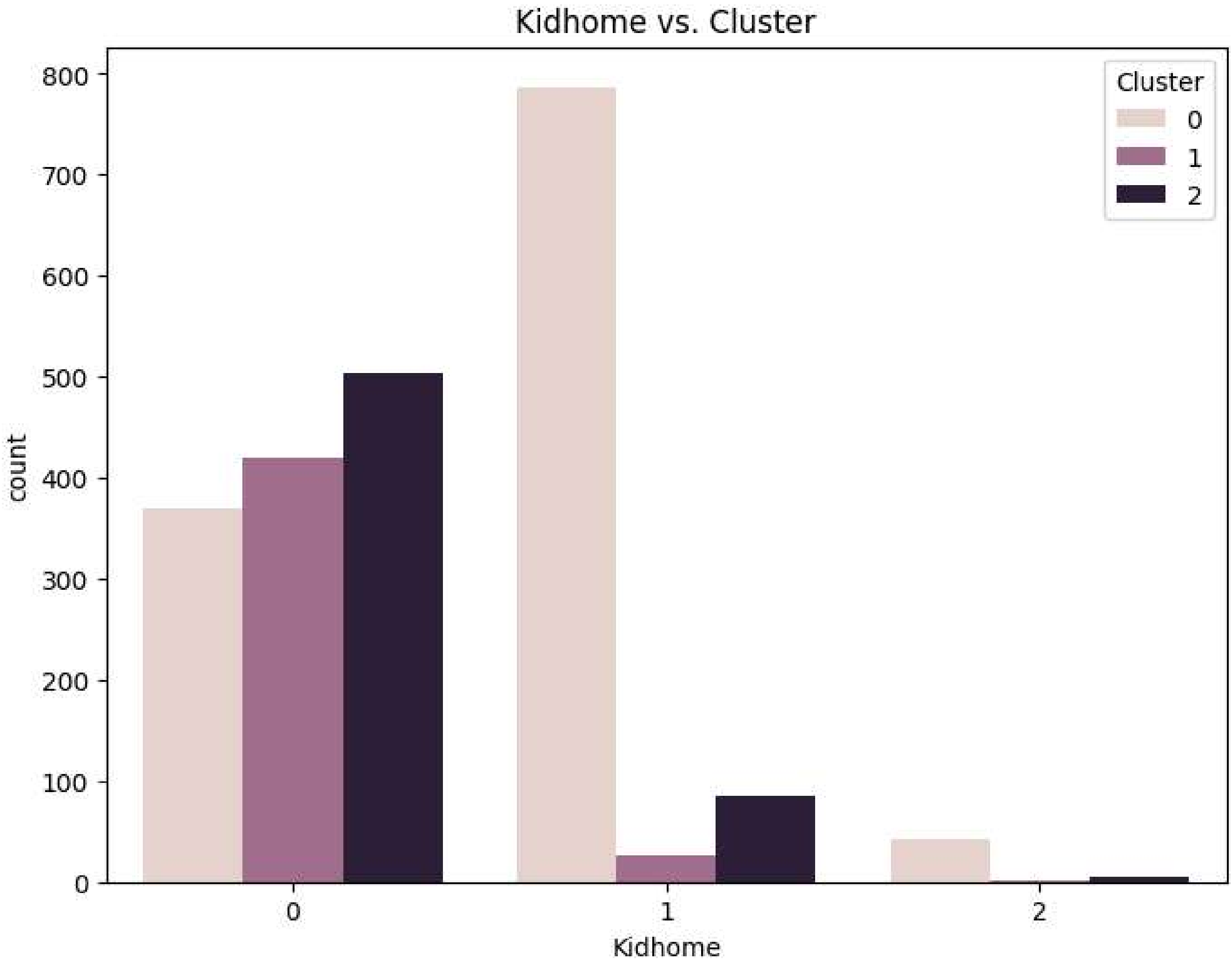
Analyze the relationship between cluster and marital status

```
plt.figure(figsize=(8, 6))
sns.countplot(x='Marital_Status', hue='Cluster', data=data)
plt.title('Marital Status vs. Cluster')
plt.show()
```

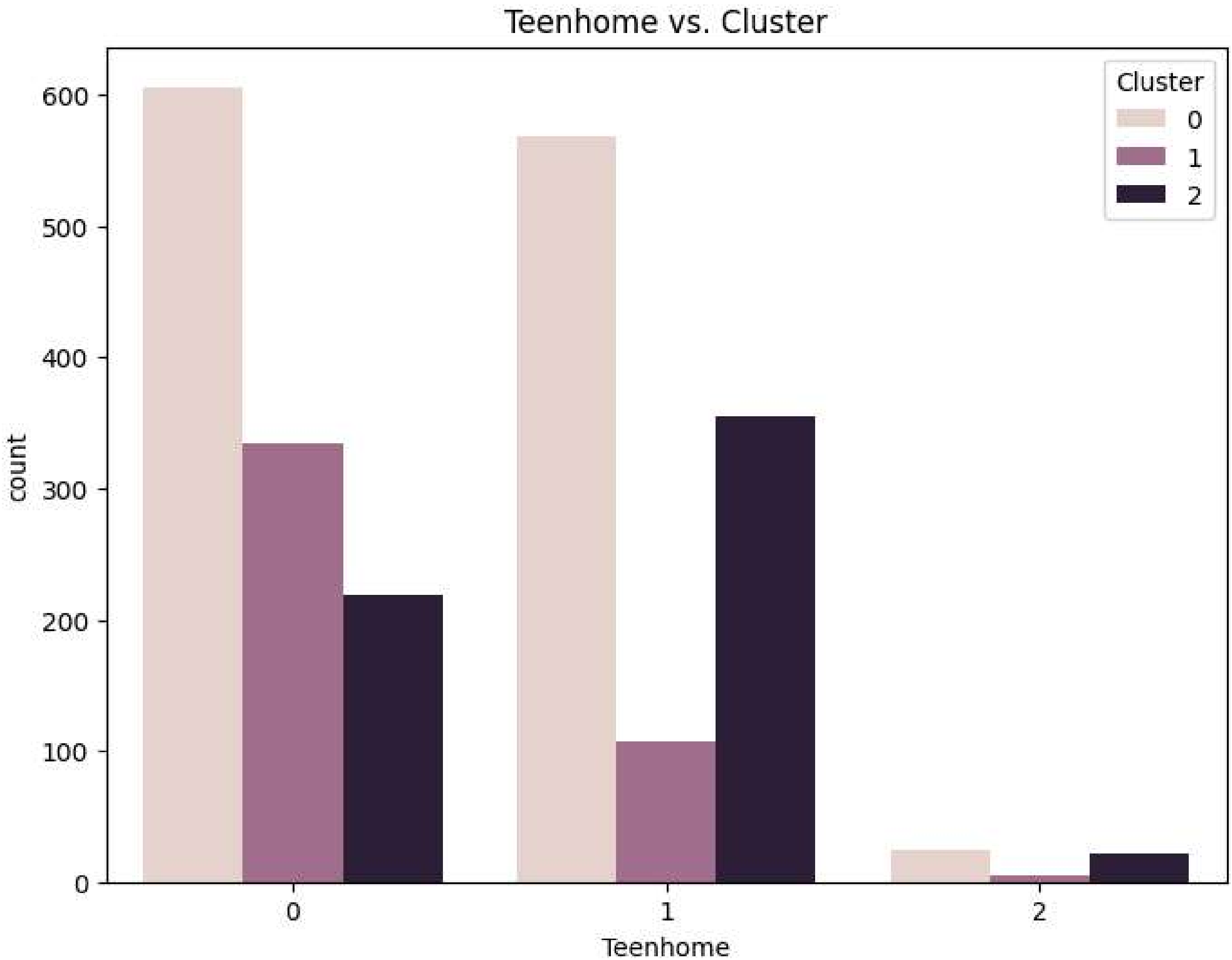


Analyze the relationship between cluster and kid/teenhome

```
plt.figure(figsize=(8, 6))
sns.countplot(x='Kidhome', hue='Cluster', data=data)
plt.title('Kidhome vs. Cluster')
plt.show()
```

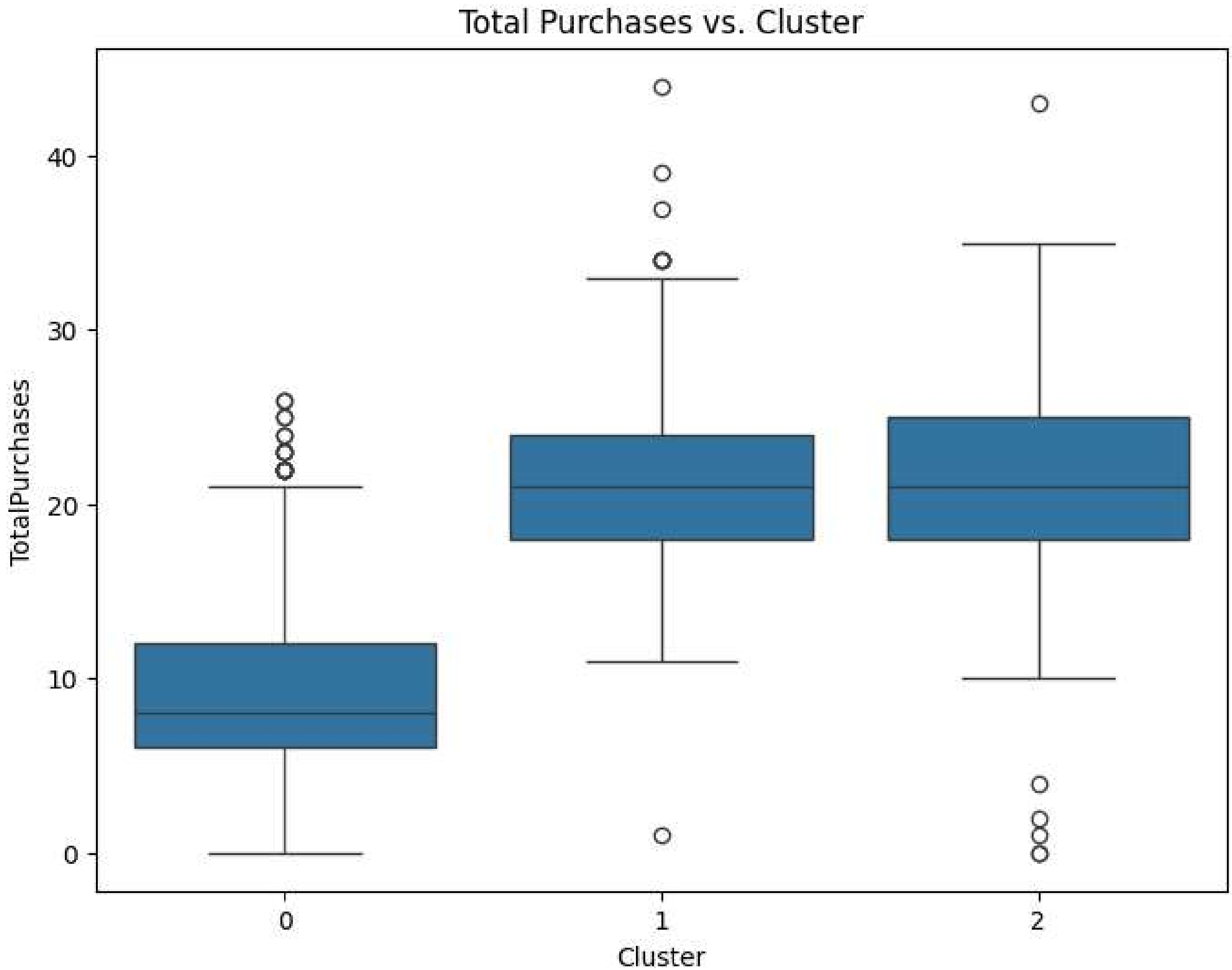



```
plt.figure(figsize=(8, 6))
sns.countplot(x='Teenhome', hue='Cluster', data=data)
plt.title('Teenhome vs. Cluster')
plt.show()
```



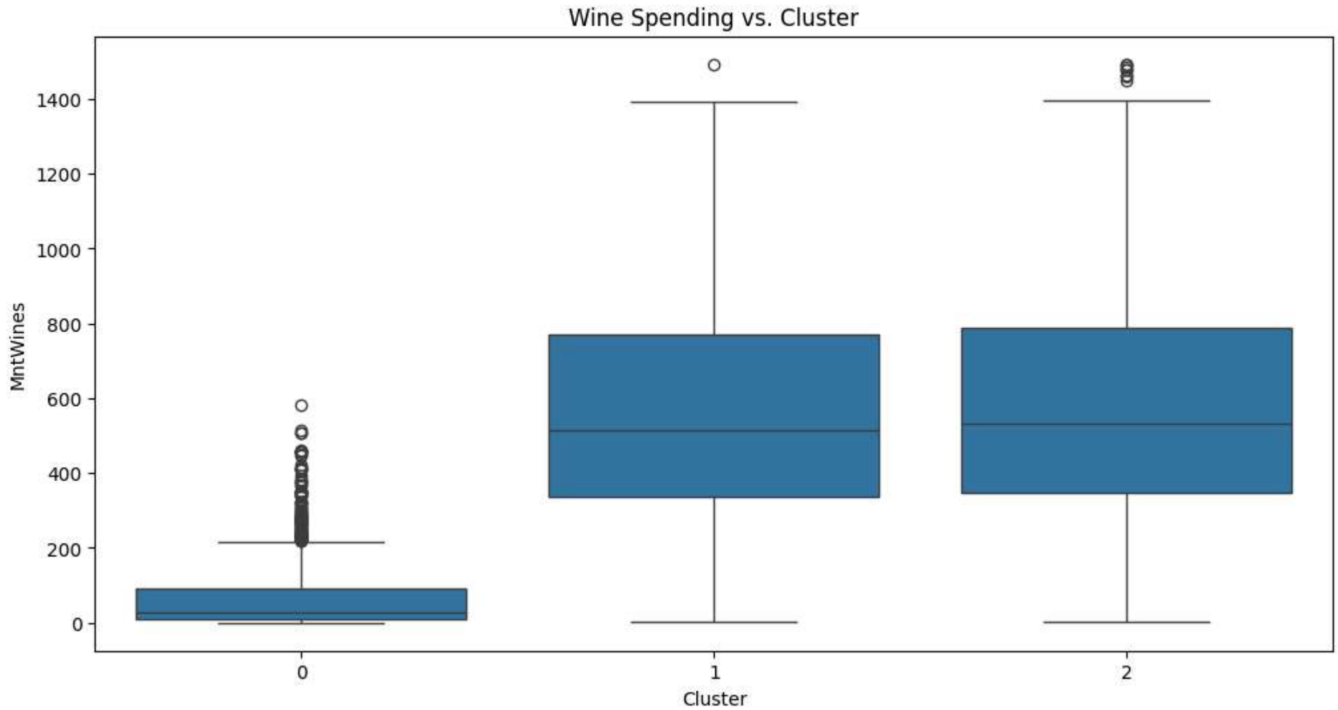
Analyze the relationship between cluster and total purchases

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Cluster', y='TotalPurchases', data=data)
plt.title('Total Purchases vs. Cluster')
plt.show()
```

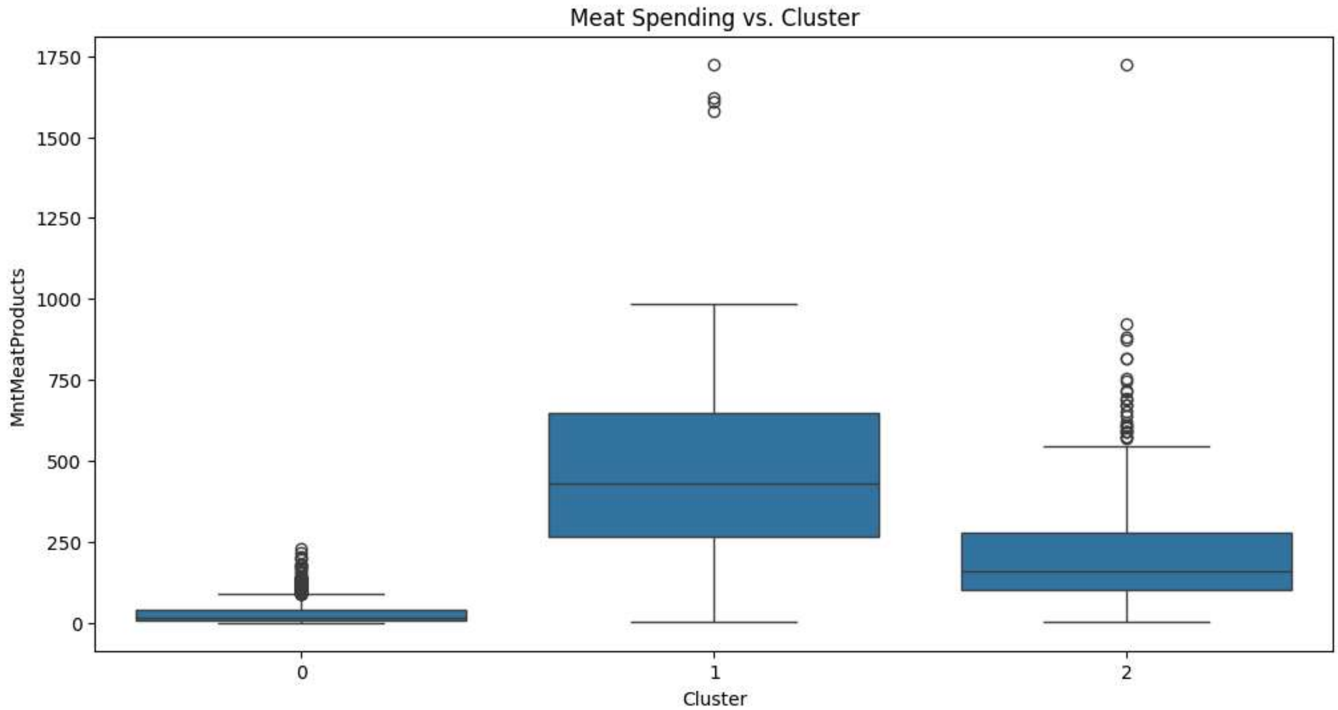


Analyze the relationship between cluster and different product categories

```
plt.figure(figsize=(12, 6))
sns.boxplot(x='Cluster', y='MntWines', data=data)
plt.title('Wine Spending vs. Cluster')
plt.show()
```

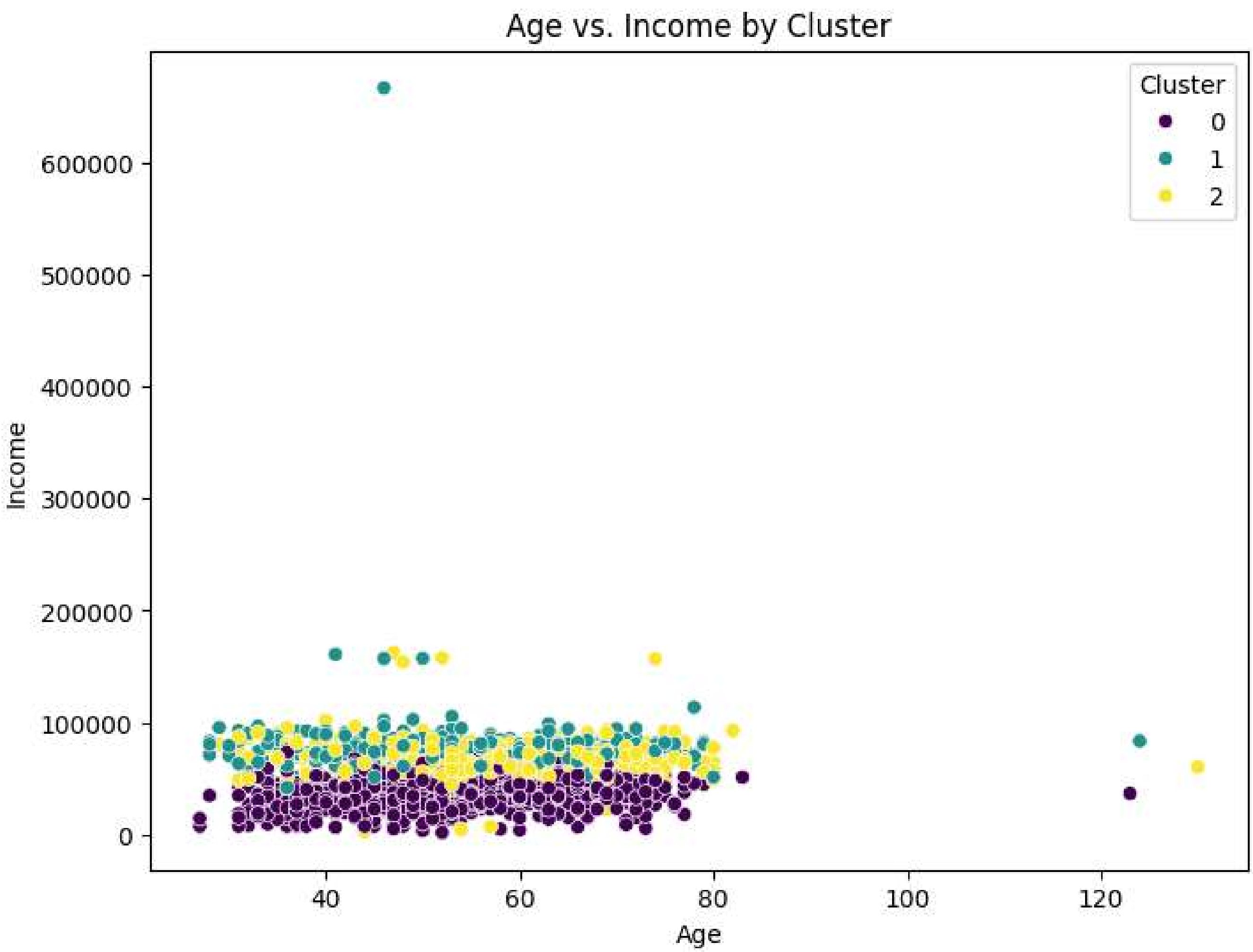



```
plt.figure(figsize=(12, 6))
sns.boxplot(x='Cluster', y='MntMeatProducts', data=data)
plt.title('Meat Spending vs. Cluster')
plt.show()
```



Scatter plot of age vs. income, color-coded by cluster

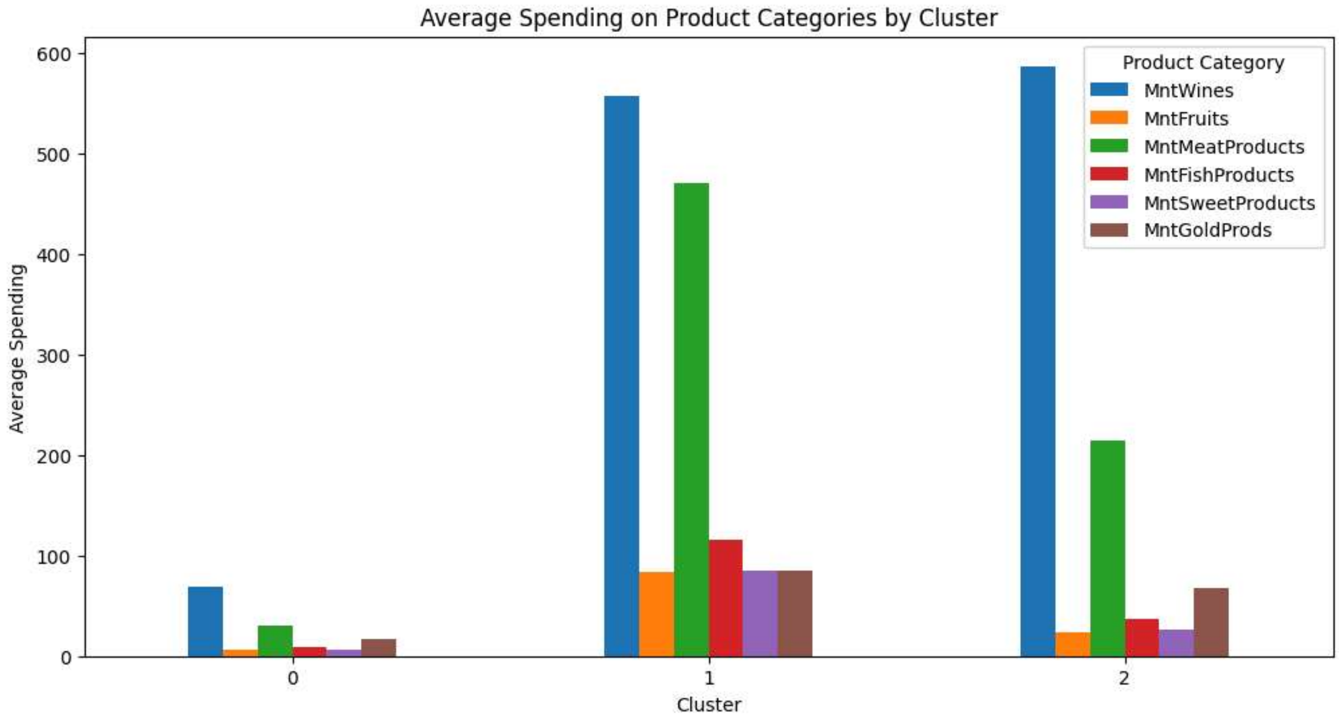
```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Age', y='Income', hue='Cluster', data=data, palette='viridis')
plt.title('Age vs. Income by Cluster')
plt.show()
```



Bar chart showing average spending on different product categories by cluster

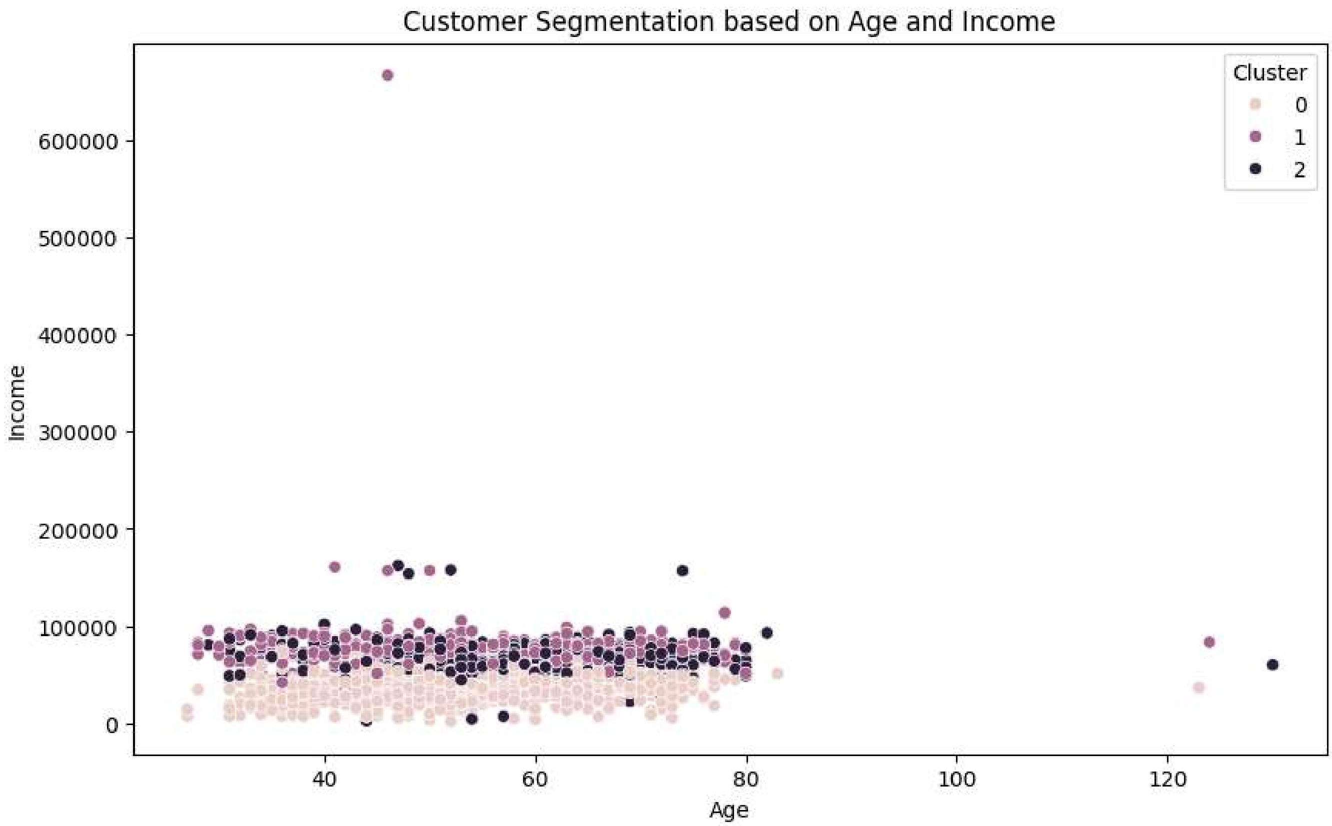
```
product_categories = ['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSwe
average_spending_by_cluster = data.groupby('Cluster')[product_categories].mean()

average_spending_by_cluster.plot(kind='bar', figsize=(12, 6))
plt.title('Average Spending on Product Categories by Cluster')
plt.xlabel('Cluster')
plt.ylabel('Average Spending')
plt.xticks(rotation=0)
plt.legend(title='Product Category')
plt.show()
```

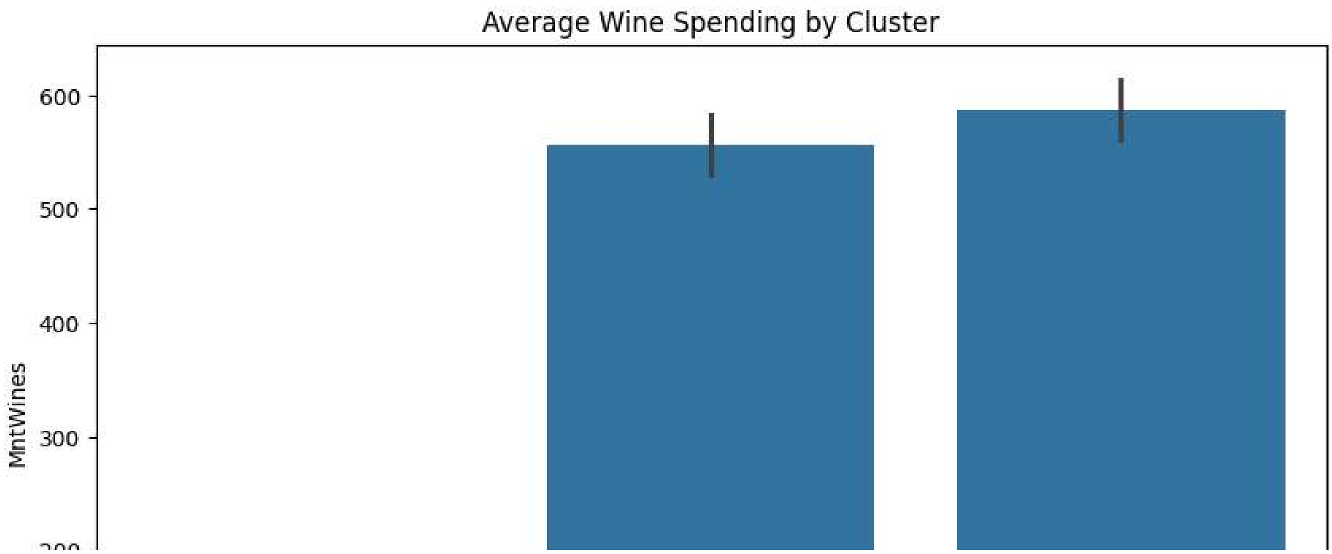


Histograms of age, income, and other variables by cluster

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Age', y='Income', hue='Cluster', data=data)
plt.title('Customer Segmentation based on Age and Income')
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.barplot(x='Cluster', y='MntWines', data=data)
plt.title('Average Wine Spending by Cluster')
plt.show()
```



Customer Lifetime Value (CLTV) Analysis

Calculate average purchase value for each customer