

```
#Importing required libraries
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import pointbiseerialr

import requests
import pandas as pd
import io

url = 'https://github.com/Praveenraj0803/OIBSIP/blob/main/Customer%20Segmentation%20Analysis/ifood_df%20(1).csv'
response = requests.get(url)

data = pd.read_csv(io.StringIO(response.text))

data.head()
```



	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	...	marit
0	58138.0	0	0	58	635	88	546	172	88	88	...	
1	46344.0	1	1	38	11	1	6	2	1	6	...	
2	71613.0	0	0	26	426	49	127	111	21	42	...	
3	26646.0	1	0	26	11	4	20	10	3	5	...	
4	58293.0	1	0	94	173	43	118	46	27	15	...	

5 rows × 39 columns



```
data.columns

Index(['Income', 'Kidhome', 'Teenhome', 'Recency', 'MntWines', 'MntFruits',
      'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
      'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
      'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
      'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
      'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue', 'Response',
      'Age', 'Customer_Days', 'marital_Divorced', 'marital_Married',
      'marital_Single', 'marital_Together', 'marital_Widow',
      'education_2n Cycle', 'education_Basic', 'education_Graduation',
      'education_Master', 'education_PhD', 'MntTotal', 'MntRegularProds',
      'AcceptedCmpOverall'],
      dtype='object')
```

```
data.isna().sum()

Income      0
Kidhome     0
Teenhome    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
Age              0
```

```

Customer_Days      0
marital_Divorced   0
marital_Married    0
marital_Single     0
marital_Together   0
marital_Widow      0
education_2n Cycle 0
education_Basic    0
education_Graduation 0
education_Master   0
education_PhD      0
MntTotal           0
MntRegularProds    0
AcceptedCmpOverall 0
dtype: int64

```

```
data.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2205 entries, 0 to 2204
Data columns (total 39 columns):
 #   Column                                Non-Null Count  Dtype
---  -
 0   Income                               2205 non-null  float64
 1   Kidhome                             2205 non-null  int64
 2   Teenhome                            2205 non-null  int64
 3   Recency                             2205 non-null  int64
 4   MntWines                            2205 non-null  int64
 5   MntFruits                           2205 non-null  int64
 6   MntMeatProducts                     2205 non-null  int64
 7   MntFishProducts                     2205 non-null  int64
 8   MntSweetProducts                    2205 non-null  int64
 9   MntGoldProds                        2205 non-null  int64
10   NumDealsPurchases                   2205 non-null  int64
11   NumWebPurchases                     2205 non-null  int64
12   NumCatalogPurchases                 2205 non-null  int64
13   NumStorePurchases                   2205 non-null  int64
14   NumWebVisitsMonth                   2205 non-null  int64
15   AcceptedCmp3                        2205 non-null  int64
16   AcceptedCmp4                        2205 non-null  int64
17   AcceptedCmp5                        2205 non-null  int64
18   AcceptedCmp1                        2205 non-null  int64
19   AcceptedCmp2                        2205 non-null  int64
20   Complain                            2205 non-null  int64
21   Z_CostContact                       2205 non-null  int64
22   Z_Revenue                           2205 non-null  int64
23   Response                            2205 non-null  int64
24   Age                                 2205 non-null  int64
25   Customer_Days                       2205 non-null  int64
26   marital_Divorced                    2205 non-null  int64
27   marital_Married                     2205 non-null  int64
28   marital_Single                      2205 non-null  int64
29   marital_Together                    2205 non-null  int64
30   marital_Widow                       2205 non-null  int64
31   education_2n Cycle                  2205 non-null  int64
32   education_Basic                     2205 non-null  int64
33   education_Graduation                 2205 non-null  int64
34   education_Master                     2205 non-null  int64
35   education_PhD                       2205 non-null  int64
36   MntTotal                            2205 non-null  int64
37   MntRegularProds                     2205 non-null  int64
38   AcceptedCmpOverall                  2205 non-null  int64
dtypes: float64(1), int64(38)
memory usage: 672.0 KB

```

```
data.nunique()
```

```

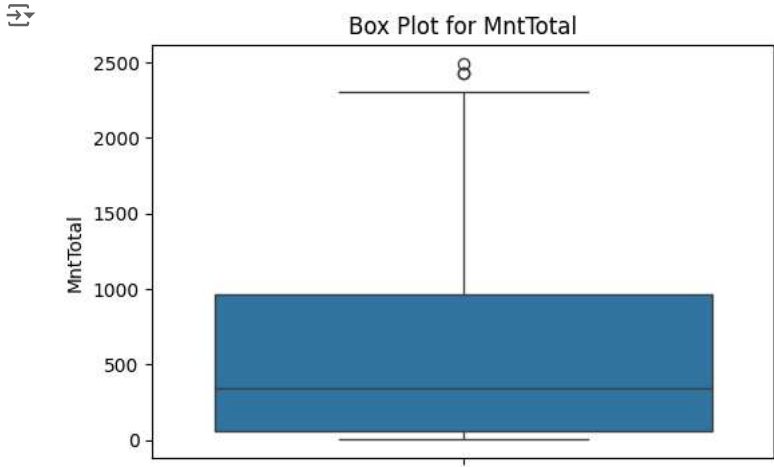
Income           1963
Kidhome          3
Teenhome         3
Recency          100
MntWines         775
MntFruits        158
MntMeatProducts  551
MntFishProducts  182
MntSweetProducts 176
MntGoldProds     212
NumDealsPurchases 15
NumWebPurchases  15
NumCatalogPurchases 13
NumStorePurchases 14
NumWebVisitsMonth 16

```

```
AcceptedCmp3      2
AcceptedCmp4      2
AcceptedCmp5      2
AcceptedCmp1      2
AcceptedCmp2      2
Complain         2
Z_CostContact     1
Z_Revenue        1
Response         2
Age             56
Customer_Days    662
marital_Divorced  2
marital_Married   2
marital_Single    2
marital_Together  2
marital_Widow     2
education_2n Cycle 2
education_Basic   2
education_Graduation 2
education_Master  2
education_PhD     2
MntTotal         897
MntRegularProds   974
AcceptedCmpOverall 5
dtype: int64
```

```
data.drop(columns=['Z_CostContact', 'Z_Revenue'],inplace=True)
```

```
plt.figure(figsize=(6, 4))
sns.boxplot(data=data, y='MntTotal')
plt.title('Box Plot for MntTotal')
plt.ylabel('MntTotal')
plt.show()
```



```
Q1 = data['MntTotal'].quantile(0.25)
Q3 = data['MntTotal'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = data[(data['MntTotal'] < lower_bound) | (data['MntTotal'] > upper_bound)]
outliers.head()
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	...	ma
1159	90638.0	0	0	29	1156	120	915	94	144	96	...	
1467	87679.0	0	0	62	1259	172	815	97	148	33	...	
1547	90638.0	0	0	29	1156	120	915	94	144	96	...	

3 rows x 37 columns

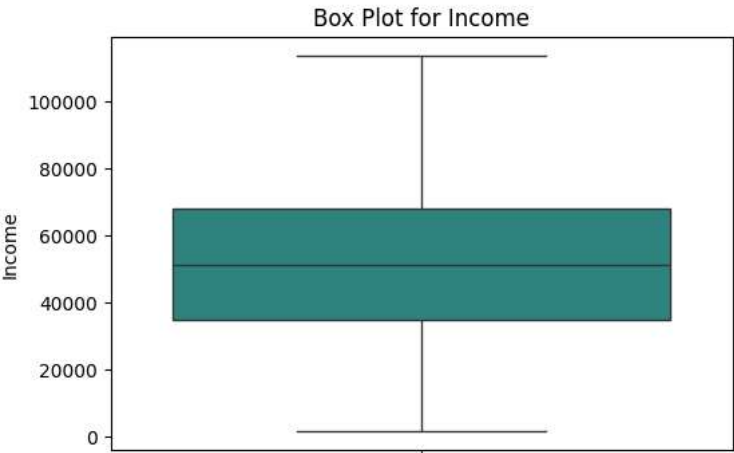
```
data = data[(data['MntTotal'] > lower_bound) & (data['MntTotal'] < upper_bound)]
data.describe()
```



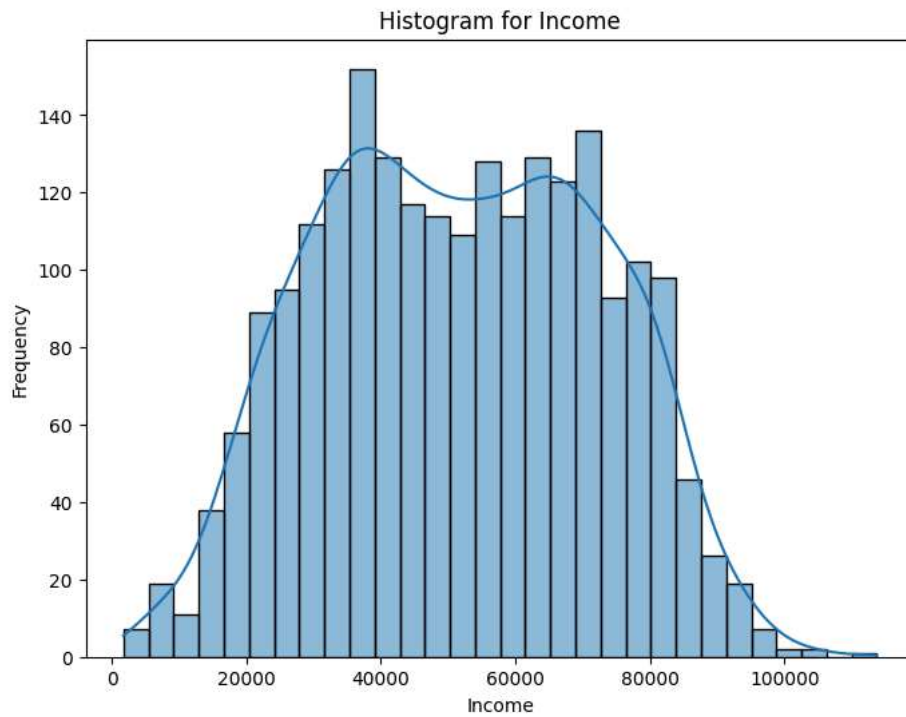
	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts
count	2202.000000	2202.000000	2202.000000	2202.000000	2202.000000	2202.000000	2202.000000	2202.000000	2202.000000
mean	51570.283379	0.442779	0.507266	49.021344	304.960036	26.252044	164.336058	37.678474	26.967302
std	20679.438848	0.537250	0.544429	28.944211	336.135586	39.589747	216.312982	54.821185	40.926101
min	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	35182.500000	0.000000	0.000000	24.000000	24.000000	2.000000	16.000000	3.000000	1.000000
50%	51258.500000	0.000000	0.000000	49.000000	176.500000	8.000000	68.000000	12.000000	8.000000
75%	68146.500000	1.000000	1.000000	74.000000	505.000000	33.000000	230.750000	50.000000	33.000000
max	113734.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	262.000000

8 rows × 37 columns

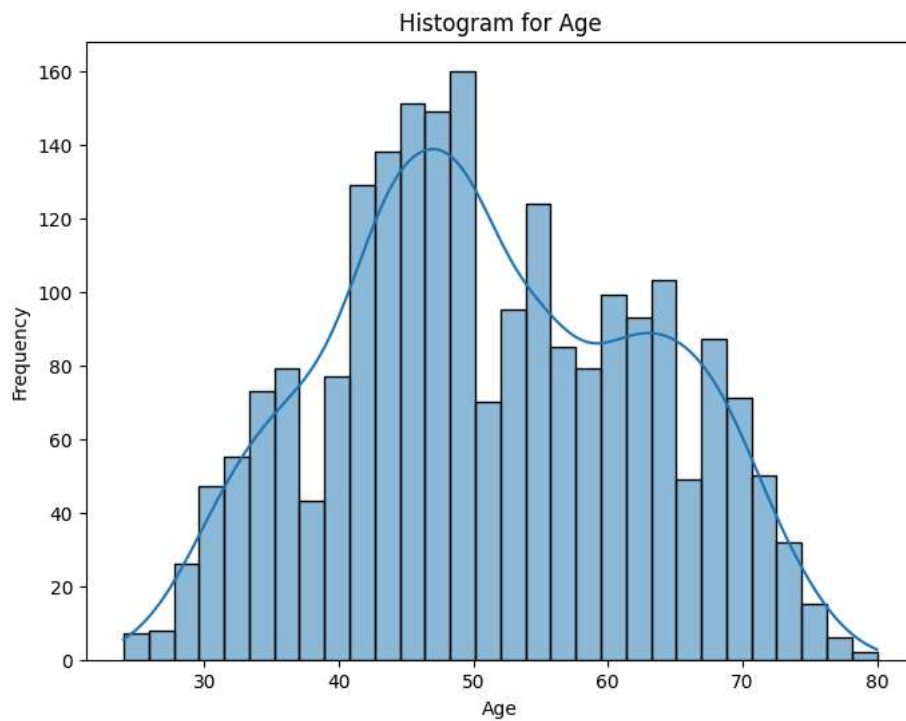
```
#Box plot and histogram for income
plt.figure(figsize=(6, 4))
sns.boxplot(data=data, y='Income', palette='viridis')
plt.title('Box Plot for Income')
plt.ylabel('Income')
plt.show()
```



```
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='Income', bins=30, kde=True)
plt.title('Histogram for Income')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.show()
```



```
#Histogram for Age
plt.figure(figsize=(8, 6))
sns.histplot(data=data, x='Age', bins=30, kde=True)
plt.title('Histogram for Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



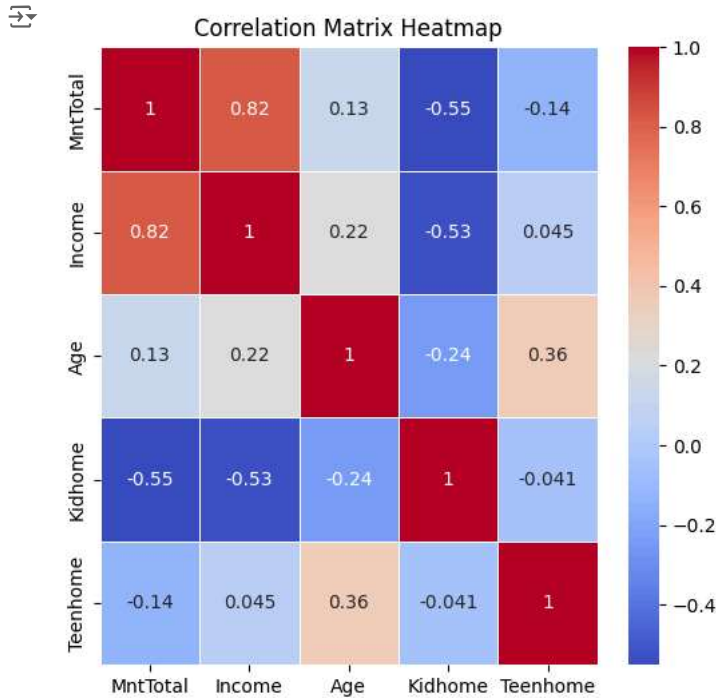
```
print("Skewness: %f" % data['Age'].skew())
print("Kurtosis: %f" % data['Age'].kurt())
```



```
Skewness: 0.091227
Kurtosis: -0.796125
```

```
cols_demographics = ['Income', 'Age']
cols_children = ['Kidhome', 'Teenhome']
cols_marital = ['marital_Divorced', 'marital_Married', 'marital_Single', 'marital_Together', 'marital_Widow']
cols_mnt = ['MntTotal', 'MntRegularProds', 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds']
cols_communication = ['Complain', 'Response', 'Customer_Days']
cols_campaigns = ['AcceptedCmpOverall', 'AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5']
cols_source_of_purchase = ['NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth']
cols_education = ['education_2n Cycle', 'education_Basic', 'education_Graduation', 'education_Master', 'education_PhD']
```

```
corr_matrix = data[['MntTotal']+cols_demographics+cols_children].corr()
plt.figure(figsize=(6,6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



```
for col in cols_marital:
    correlation, p_value = pointbserialr(data[col], data['MntTotal'])
    print(f'{correlation:.4f}: Point-Biserial Correlation for {col} with p-value {p_value:.4f}')
```

```
0.0053: Point-Biserial Correlation for marital_Divorced with p-value 0.8041
-0.0188: Point-Biserial Correlation for marital_Married with p-value 0.3767
0.0011: Point-Biserial Correlation for marital_Single with p-value 0.9571
0.0008: Point-Biserial Correlation for marital_Together with p-value 0.9708
0.0370: Point-Biserial Correlation for marital_Widow with p-value 0.0826
```

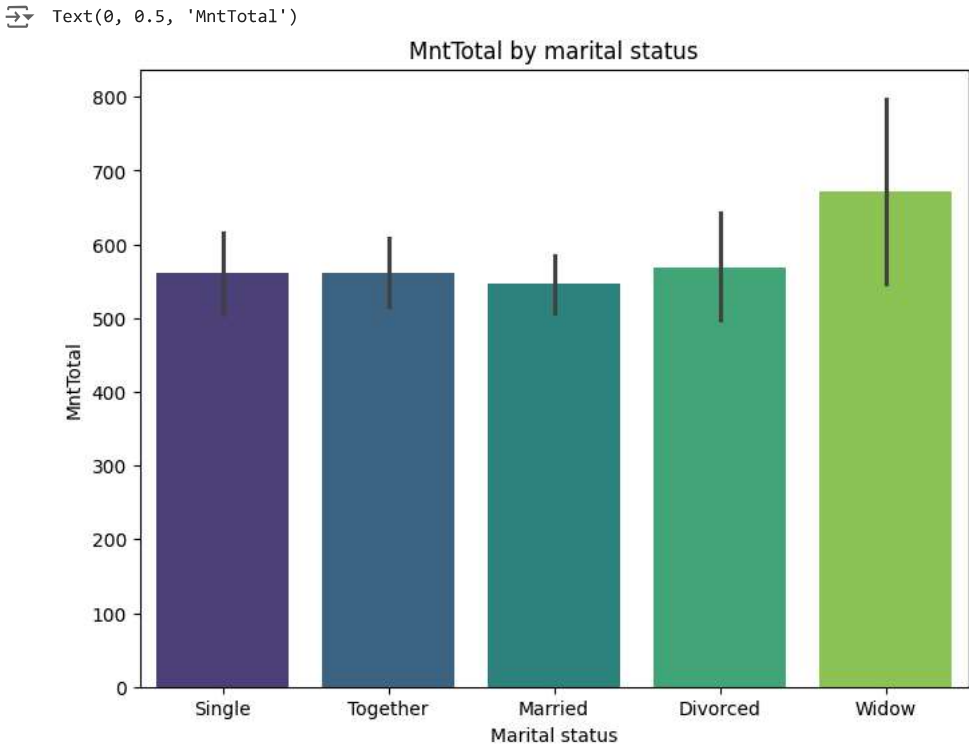
```
for col in cols_education:
    correlation, p_value = pointbserialr(data[col], data['MntTotal'])
    print(f'{correlation:.4f}: Point-Biserial Correlation for {col} with p-value {p_value:.4f}')
```

```
-0.0593: Point-Biserial Correlation for education_2n Cycle with p-value 0.0054
-0.1389: Point-Biserial Correlation for education_Basic with p-value 0.0000
0.0159: Point-Biserial Correlation for education_Graduation with p-value 0.4551
0.0004: Point-Biserial Correlation for education_Master with p-value 0.9842
0.0737: Point-Biserial Correlation for education_PhD with p-value 0.0005
```

```
def get_marital_status(row):
    if row['marital_Divorced'] == 1:
        return 'Divorced'
    elif row['marital_Married'] == 1:
        return 'Married'
    elif row['marital_Single'] == 1:
        return 'Single'
    elif row['marital_Together'] == 1:
        return 'Together'
    elif row['marital_Widow'] == 1:
        return 'Widow'
    else:
        return 'Unknown'

data['Marital'] = data.apply(get_marital_status, axis=1)

plt.figure(figsize=(8, 6))
sns.barplot(x='Marital', y='MntTotal', data=data, palette='viridis')
plt.title('MntTotal by marital status')
plt.xlabel('Marital status')
plt.ylabel('MntTotal')
```



```
def get_relationship(row):
    if row['marital_Married'] == 1:
        return 1
    elif row['marital_Together'] == 1:
        return 1
    else:
        return 0

data['In_relationship'] = data.apply(get_relationship, axis=1)
data.head()
```

↗

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	...	educa
0	58138.0	0	0	58	635	88	546	172	88	88	...	
1	46344.0	1	1	38	11	1	6	2	1	6	...	
2	71613.0	0	0	26	426	49	127	111	21	42	...	
3	26646.0	1	0	26	11	4	20	10	3	5	...	
4	58293.0	1	0	94	173	43	118	46	27	15	...	

5 rows × 39 columns

```
from sklearn.cluster import KMeans
```

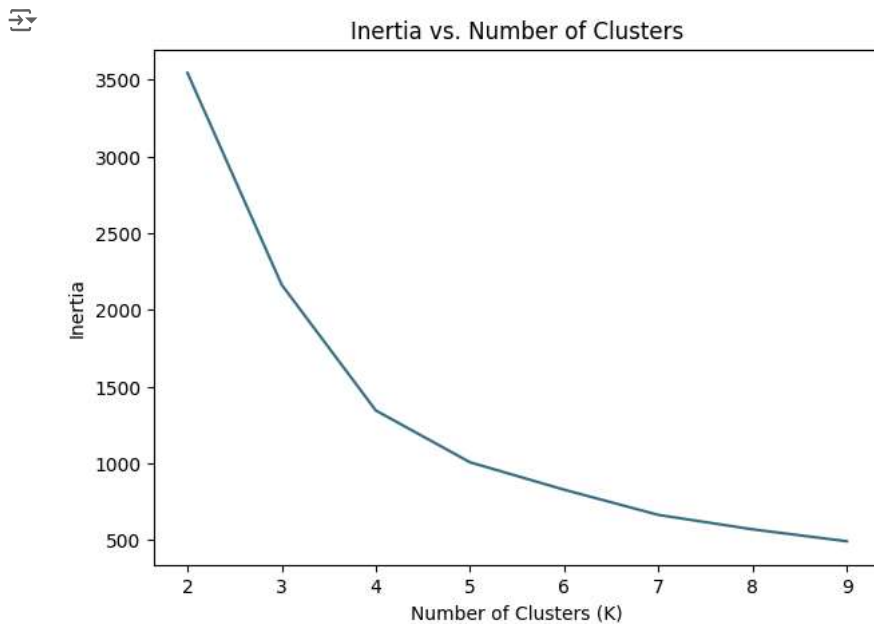
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
cols_for_clustering = ['Income', 'MntTotal', 'In_relationship']
data_scaled = data.copy()
data_scaled[cols_for_clustering] = scaler.fit_transform(data[cols_for_clustering])
data_scaled[cols_for_clustering].describe()
```

```
↗
```

	Income	MntTotal	In_relationship
count	2.202000e+03	2.202000e+03	2.202000e+03
mean	2.742785e-17	-8.873717e-17	-4.678869e-17
std	1.000227e+00	1.000227e+00	1.000227e+00
min	-2.410685e+00	-9.724232e-01	-1.348874e+00
25%	-7.926475e-01	-8.815089e-01	-1.348874e+00
50%	-1.508040e-02	-3.806058e-01	7.413589e-01
75%	8.017617e-01	7.024988e-01	7.413589e-01
max	3.006747e+00	3.048788e+00	7.413589e-01

```
from sklearn import decomposition
pca = decomposition.PCA(n_components = 2)
pca_res = pca.fit_transform(data_scaled[cols_for_clustering])
data_scaled['pc1'] = pca_res[:,0]
data_scaled['pc2'] = pca_res[:,1]
X = data_scaled[cols_for_clustering]
inertia_list = []
for K in range(2,10):
    inertia = KMeans(n_clusters=K, random_state=7).fit(X).inertia_
    inertia_list.append(inertia)

plt.figure(figsize=[7,5])
plt.plot(range(2,10), inertia_list, color=(54 / 255, 113 / 255, 130 / 255))
plt.title("Inertia vs. Number of Clusters")
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Inertia")
plt.show()
```



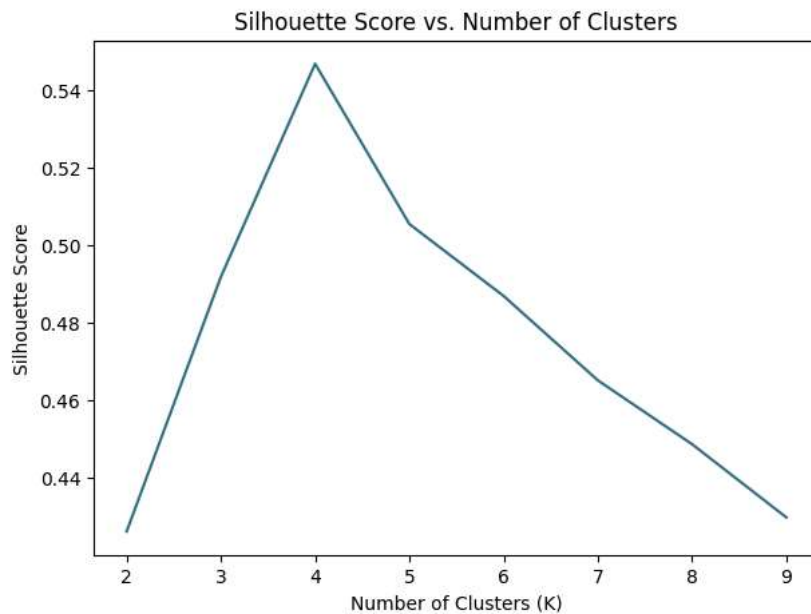


```

from sklearn.metrics import silhouette_score
silhouette_list = []
for K in range(2,10):
    model = KMeans(n_clusters = K, random_state=7)
    clusters = model.fit_predict(X)
    s_avg = silhouette_score(X, clusters)
    silhouette_list.append(s_avg)

plt.figure(figsize=[7,5])
plt.plot(range(2,10), silhouette_list, color=(54 / 255, 113 / 255, 130 / 255))
plt.title("Silhouette Score vs. Number of Clusters")
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Silhouette Score")
plt.show()

```



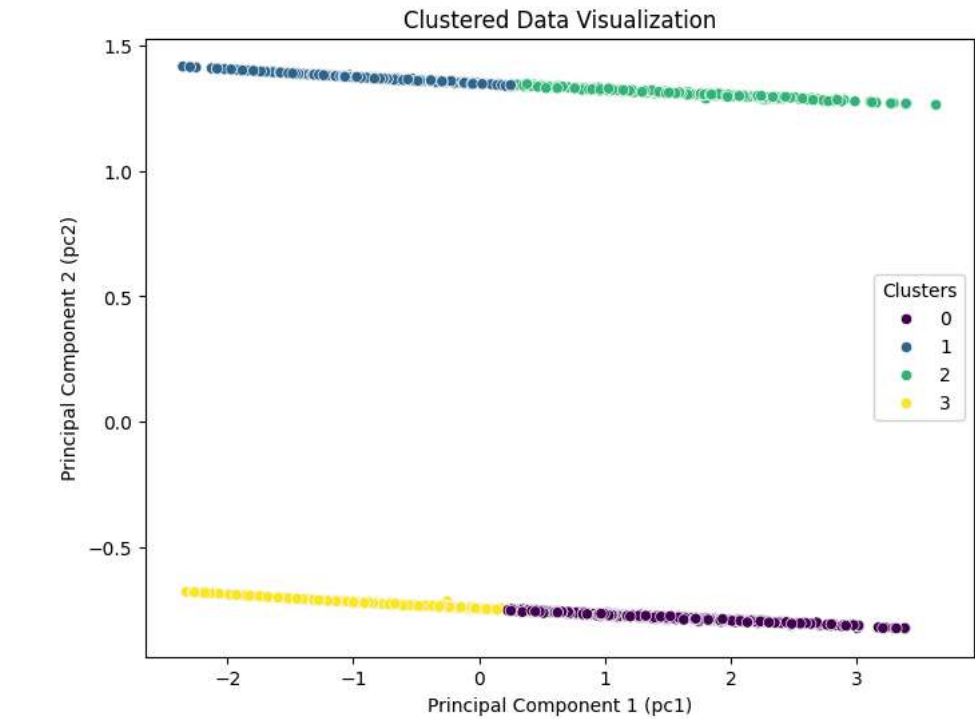
```

model = KMeans(n_clusters=4, random_state = 7)
model.fit(data_scaled[cols_for_clustering])
data_scaled['Cluster'] = model.predict(data_scaled[cols_for_clustering])

plt.figure(figsize=(8, 6))
sns.scatterplot(x='pc1', y='pc2', data=data_scaled, hue='Cluster', palette='viridis')
plt.title('Clustered Data Visualization')
plt.xlabel('Principal Component 1 (pc1)')
plt.ylabel('Principal Component 2 (pc2)')
plt.legend(title='Clusters')

```

```
<matplotlib.legend.Legend at 0x79fc9b13f490>
```



```
data['Cluster'] = data_scaled.Cluster
data.groupby('Cluster')[cols_for_clustering].mean()
```

	Income	MntTotal	In_relationship
Cluster			
0	71818.929329	1147.372792	1.0
1	37332.339956	150.761589	0.0
2	71946.155488	1159.612805	0.0
3	37892.819883	158.463158	1.0

```
mnt_data = data.groupby('Cluster')[cols_mnt].mean().reset_index()
mnt_data.head()
```

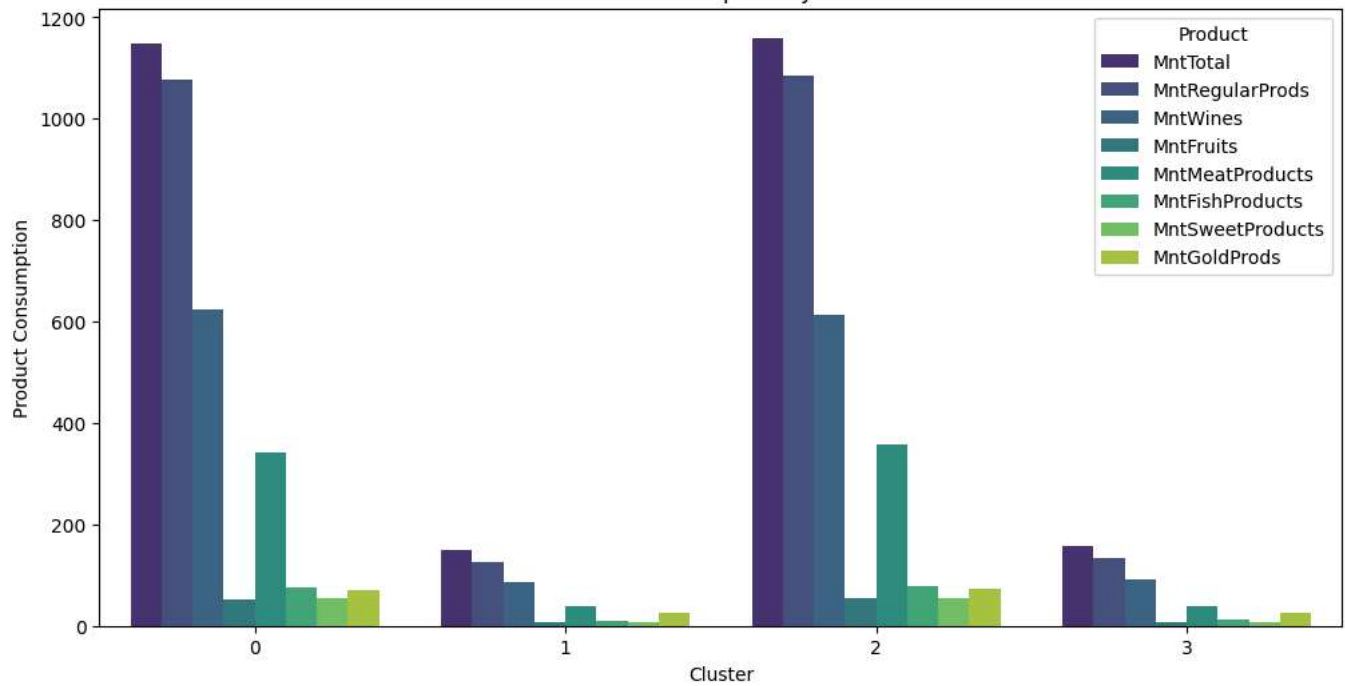
	Cluster	MntTotal	MntRegularProds	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds
0	0	1147.372792	1076.279152	623.261484	52.489399	341.326855	75.577739	54.717314	71.093640
1	1	150.761589	125.662252	85.450331	7.832230	38.774834	10.971302	7.732892	25.099338
2	2	1159.612805	1085.332317	613.862805	54.929878	357.902439	77.603659	55.314024	74.280488
3	3	158.463158	133.962573	92.046784	7.640936	39.438596	11.423392	7.913450	24.500585

```
melted_data = pd.melt(mnt_data, id_vars="Cluster", var_name="Product", value_name="Consumption")
plt.figure(figsize=(12, 6))
sns.barplot(x="Cluster", y="Consumption", hue="Product", data=melted_data, ci=None, palette="viridis")
plt.title("Product Consumption by Cluster")
plt.xlabel("Cluster")
plt.ylabel("Product Consumption")
plt.xticks(rotation=0)
plt.legend(title="Product", loc="upper right")

plt.show()
```



Product Consumption by Cluster

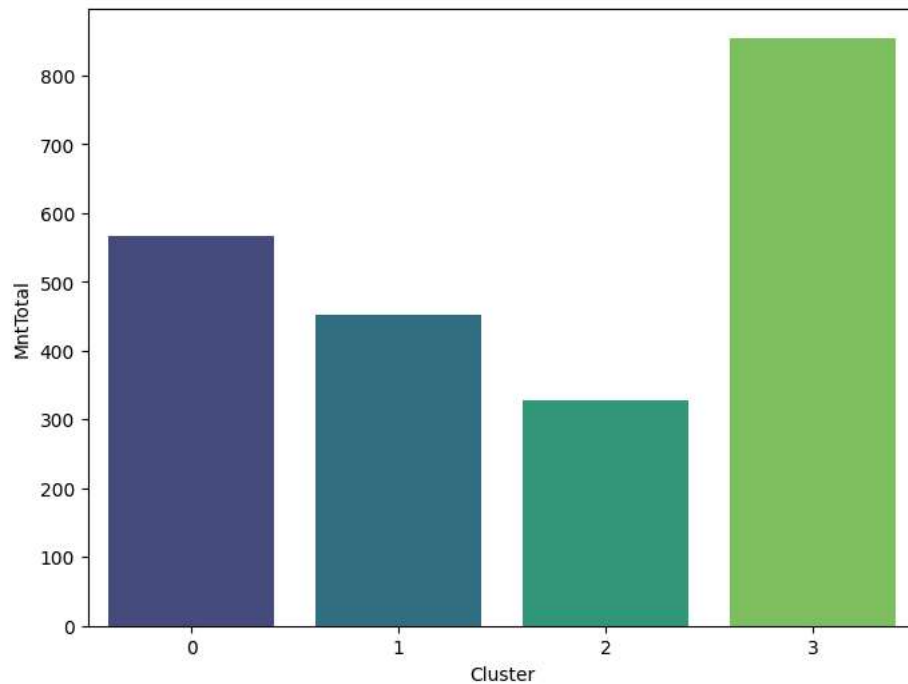


```
cluster_sizes = data.groupby('Cluster')[['MntTotal']].count().reset_index()
plt.figure(figsize=(8,6))
sns.barplot(x='Cluster', y='MntTotal', data=cluster_sizes, palette = 'viridis')
plt.title('Cluster sizes')
plt.xlabel('Cluster')
plt.ylabel('MntTotal')
```



Text(0, 0.5, 'MntTotal')

Cluster sizes

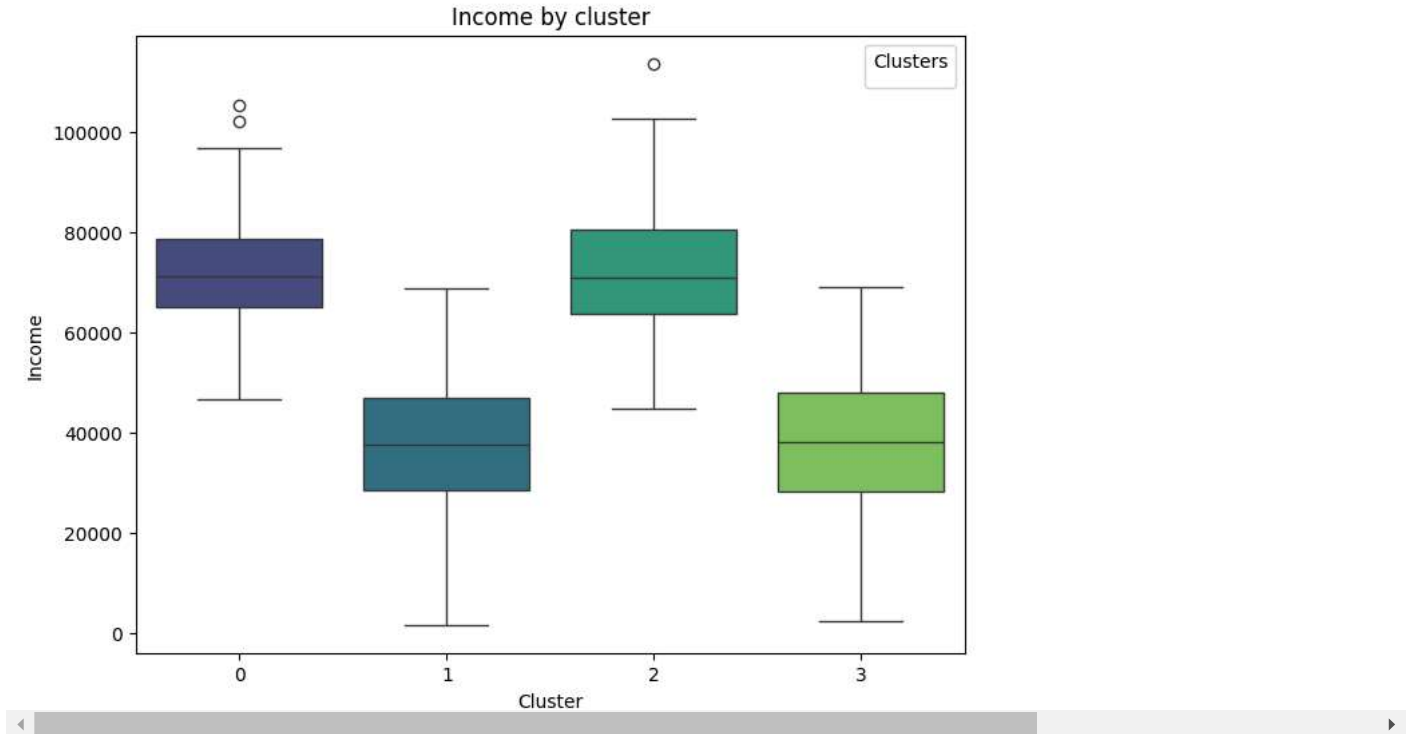


```
total_rows = len(data)
cluster_sizes['Share%'] = round(cluster_sizes['MntTotal'] / total_rows*100,0)
cluster_sizes.head()
```

	Cluster	MntTotal	Share%
0	0	566	26.0
1	1	453	21.0

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Cluster', y='Income', data=data, palette='viridis')
plt.title('Income by cluster')
plt.xlabel('Cluster')
plt.ylabel('Income')
plt.legend(title='Clusters')
```

WARNING:matplotlib.legend.No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored  
<matplotlib.legend.Legend at 0x79fc983a6980>



```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Income', y='MntTotal', data=data, hue = 'Cluster', palette='viridis')
plt.title('Income by cluster')
plt.xlabel('Income')
plt.ylabel('MntTotal')
plt.legend(title='Clusters')
```

<matplotlib.legend.Legend at 0x79fc98fdfac0>

