In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries instal
led
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/d
ocker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will lis
t all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that ge
ts preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved
outside of the current session
```

```
/kaggle/input/customer-personality-analysis/marketing_campaign.csv
```

This dataset contains various details about a company's customers. Analyzing the customers allows a company to figure out what buying habits they have and offer products or services that cater to the right customer. It allows a company to segment their customers and identify what segments they want to focus on, which then gives them the information to be more efficient with their oeprations. The goal of this proejct is to

- Explore the data
- Clean up any null values
- Visualize the data and find relationships
- Create new feautures
- Cluster the customers based on certain features and create customer profiles

In [2]:
```
# data analysis and wrangling
import pandas as pd
import numpy as np
import random as rnd

# visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Clustering
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import AgglomerativeClustering
```

In [3]:
```
df = pd.read_csv("../input/customer-personality-analysis/marketing_campaign.csv", sep="\t")
```

Overview of the data:

I want to first examine the data and figure out what each columns means and how they relate to each other. I am going to be taking a look at their datatypes and check to see if there are any null values that I will have to cleanup.

Attributes

People

ID: Customer's unique identifier

Year_Birth: Customer's birth year

Education: Customer's education level

Marital_Status: Customer's marital status

Income: Customer's yearly household income

Kidhome: Number of children in customer's household

Teenhome: Number of teenagers in customer's household

Dt_Customer: Date of customer's enrollment with the company

Recency: Number of days since customer's last purchase

Complain: 1 if the customer complained in the last 2 years, 0 otherwise

Products

MntWines: Amount spent on wine in last 2 years

MntFruits: Amount spent on fruits in last 2 years

MntMeatProducts: Amount spent on meat in last 2 years

MntFishProducts: Amount spent on fish in last 2 years

MntSweetProducts: Amount spent on sweets in last 2 years

MntGoldProds: Amount spent on gold in last 2 years

Promotion

NumDealsPurchases: Number of purchases made with a discount

AcceptedCmp1: 1 if customer accepted the offer in the 1st campaign, 0 otherwise

AcceptedCmp2: 1 if customer accepted the offer in the 2nd campaign, 0 otherwise

AcceptedCmp3: 1 if customer accepted the offer in the 3rd campaign, 0 otherwise

AcceptedCmp4: 1 if customer accepted the offer in the 4th campaign, 0 otherwise

AcceptedCmp5: 1 if customer accepted the offer in the 5th campaign, 0 otherwise

Response: 1 if customer accepted the offer in the last campaign, 0 otherwise

Place

NumWebPurchases: Number of purchases made through the company's website

NumCatalogPurchases: Number of purchases made using a catalogue

NumStorePurchases: Number of purchases made directly in stores

NumWebVisitsMonth: Number of visits to company's website in the last month

In [4]:
```
df.head()
```

Out[4]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Rec
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	

5 rows × 29 columns

In [5]:
```
# check data type of all columns
df.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
```

Data Cleaning

Luckily it seems that only the income column has null values, but asides from that the only other thing that we will have to change is to convert the Dt_customer column to a datetime type rather than an object type

In [6]:
```
# check which columns have null values, it seems only income has null values
df.isnull().sum()
```

Out[6]:
```
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
```

In [7]:
```
# isolate null values to see if there is any relationship, but it doesnt seem
 like there is
null_income = df.loc[df.Income.isnull()]
```

In [8]:
```
# fill in null income with the median income
med = df.Income.median()
df.Income.fillna(med, inplace=True)
df.isnull().sum()
```

Out[8]:
```
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
```

In [9]:
```
#convert the column to datetime format
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], infer_datetime_format = True)
```

Data Exploration and Visualization

With the null values filled in and the columns in their appropriate datatype we can now begin exploring the data and forming features. I will create some new features and create some graphs inbetween to visualize relationships and see what other features I might have to create.

In [10]:
```
#created an age column
df['Age']=2021-df['Year_Birth']
```

In [11]:
```
#created a column showing the total amount a customer spent
df['TotalMnt'] = df.MntMeatProducts + df. MntWines + df.MntFruits + df.MntFish
Products + df.MntSweetProducts + df.MntGoldProds
```

In [12]:
```
#combined the kidhome and teenhome columns into one column that shows the numb
er of children per household
df['children'] = df.Kidhome + df.Teenhome
```

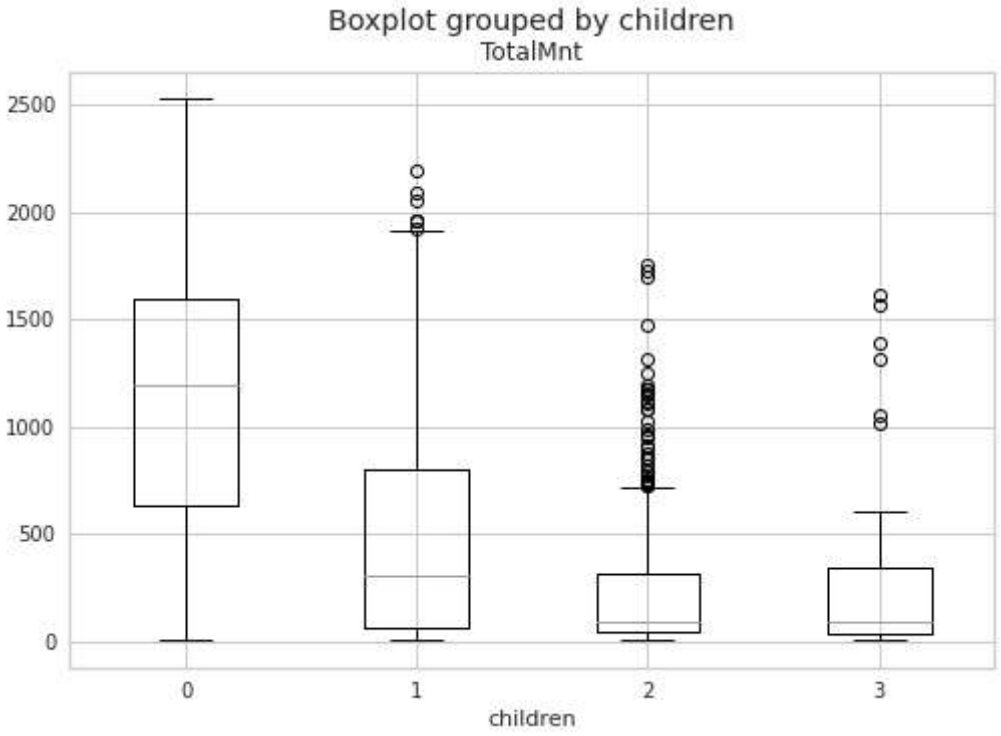
In [13]:
```
#checks how much of a customer's income is spent purchasing these products
df['Income_to_spend'] = round(df['TotalMnt']/df['Income'],3)
```

In [14]:
```
#number of purchases the customer made
df['TotalNumPurchases'] = df.NumWebPurchases + df.NumCatalogPurchases + df.Num
StorePurchases

#number of purchases that were discounted
df['num_discounted'] = round(df['NumDealsPurchases']/df['TotalNumPurchases'],3
)

#check which percent of purchases that were made per platform
df['web_to_total'] = round(df.NumWebPurchases/df.TotalNumPurchases,3)
df['catalog_to_total'] = round(df.NumCatalogPurchases/df.TotalNumPurchases,3)
df['Store_to_total'] = round(df.NumStorePurchases/df.TotalNumPurchases,3)

#fill the na values with 0 because if it is null then that means that there we
re no purchases for that customer, through that specific channel
df.fillna(0, inplace=True)
```

In [15]:
```
#counts the number of campaigns that the customer accepted
df['num_cmp'] = df.AcceptedCmp1 + df.AcceptedCmp2 + df.AcceptedCmp3 + df.Accep
tedCmp4 + df.AcceptedCmp5 + df.Response
```

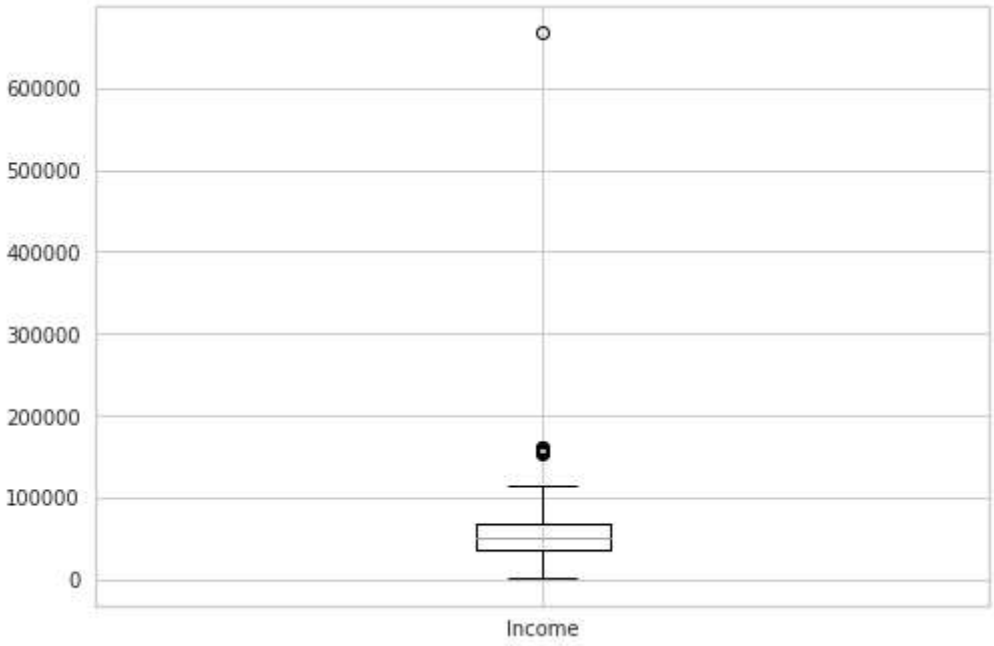
In [16]:
```
#graph to see if there is a relationship between total amount spent and number of children
df.boxplot(column = 'TotalMnt', by = 'children')
```

Out[16]: <AxesSubplot:title={'center':'TotalMnt'}, xlabel='children'>



In [17]:
```
#graph to spot outliers
df.boxplot(column = 'Income')
```

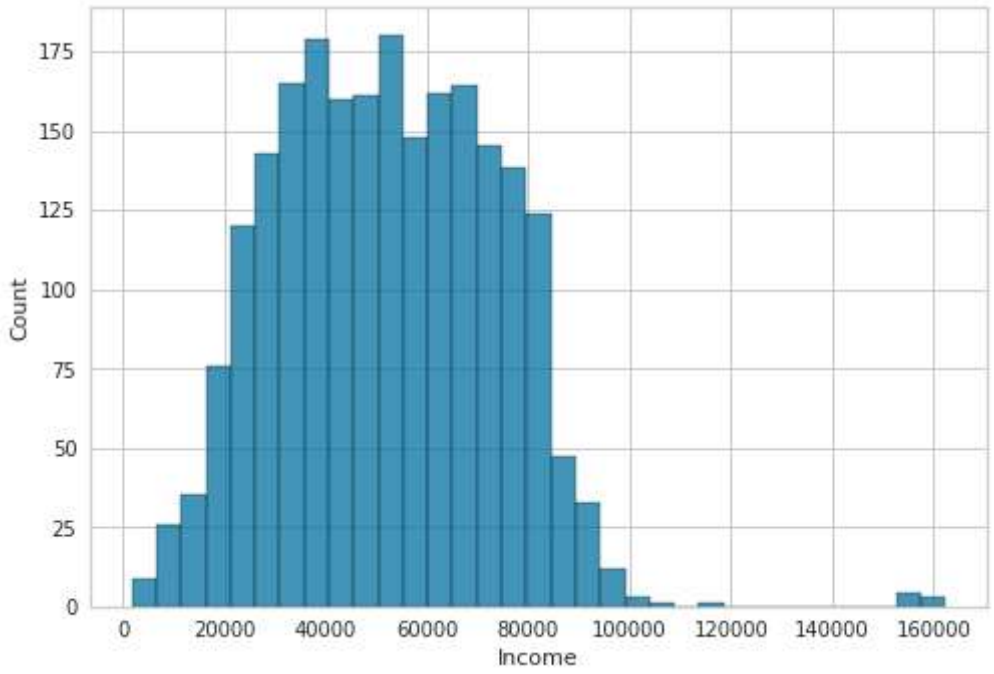
Out[17]: <AxesSubplot:>



In [18]:
```
#removed the one big outlier spotted
df=df.loc[df.Income<df.Income.max()]
```

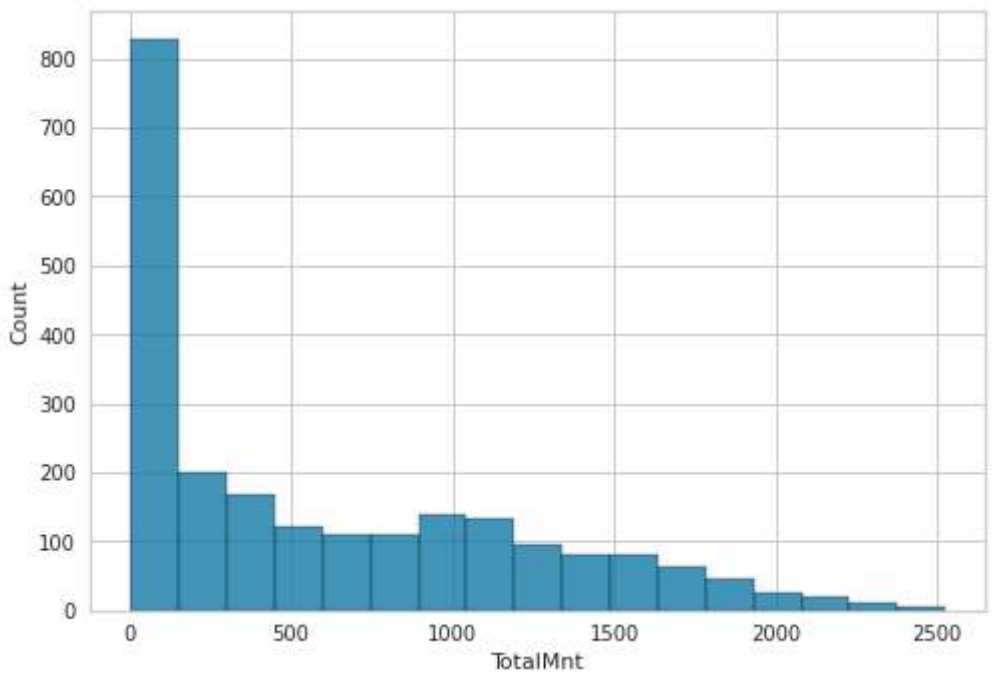
In [19]:
```
#check the distribution of the income, seems relatively normal
sns.histplot(df.Income)
```

Out[19]: <AxesSubplot:xlabel='Income', ylabel='Count'>



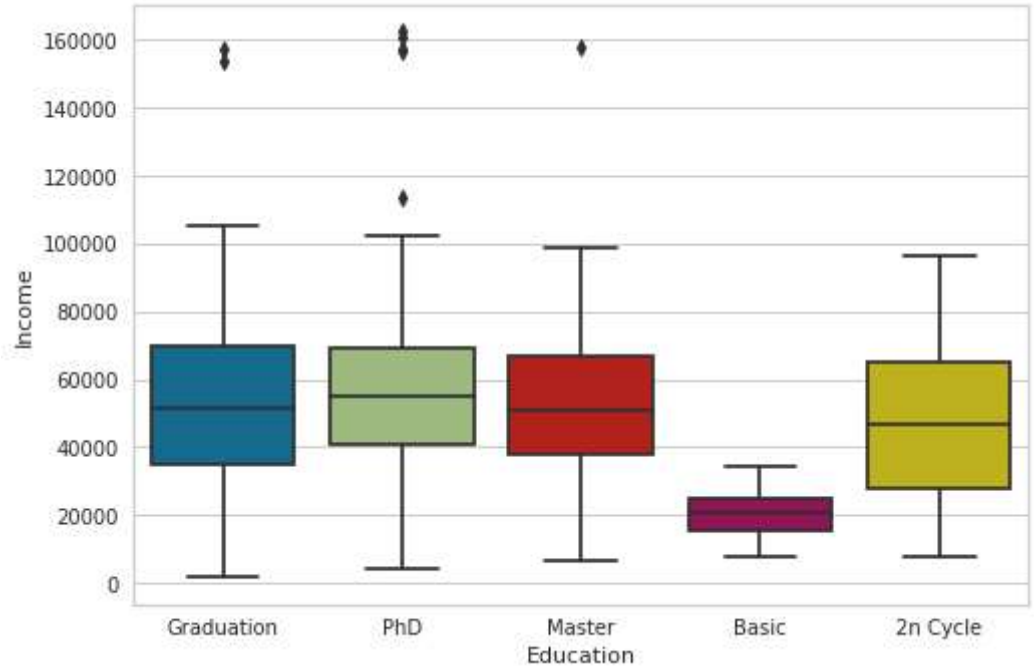
In [20]:
```
#check the distribution of amount spent and it seems skewed
sns.histplot(df.TotalMnt)
```

Out[20]: <AxesSubplot:xlabel='TotalMnt', ylabel='Count'>



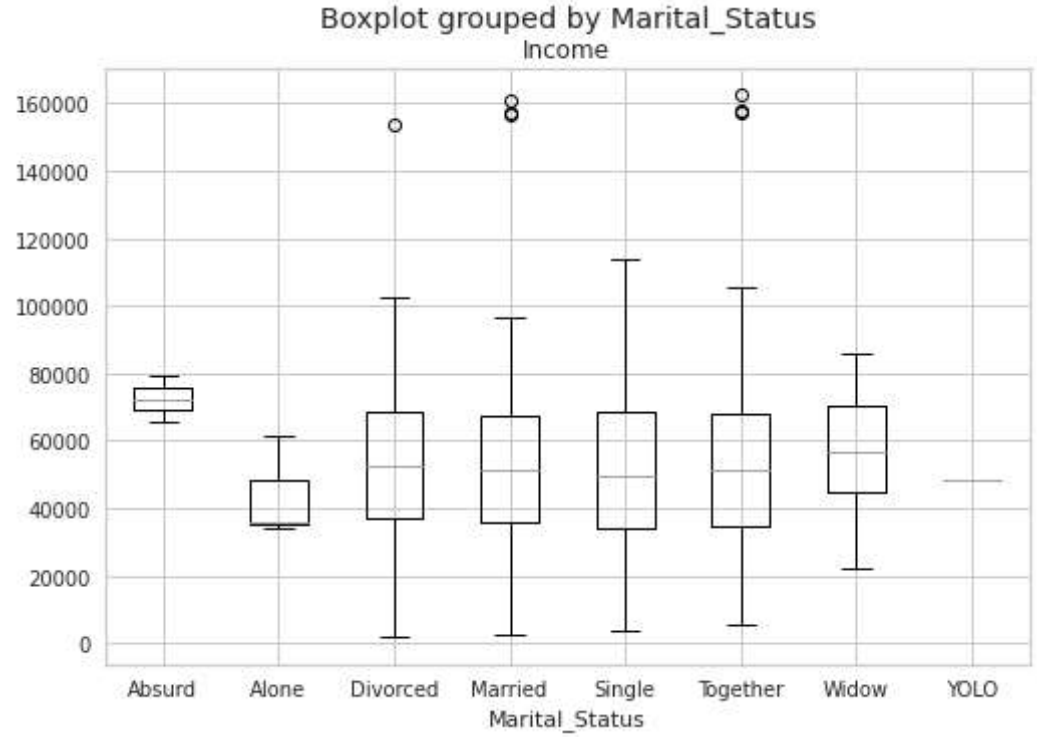
In [21]:
```
#checked to see if education had a big impact on income and aside from the bas
ic education level, it did no seem like it
sns.boxplot(x=df.Education, y=df.Income)
```

Out[21]: <AxesSubplot:xlabel='Education', ylabel='Income'>



In [22]:
```
#checked the income range between different marital_statuses
df.boxplot(column = 'Income', by = 'Marital_Status')
```

Out[22]: <AxesSubplot:title={'center':'Income'}, xlabel='Marital_Status'>



In [23]:
```
#spotted two odd entries
df.Marital_Status.unique()
```

Out[23]:
```
array(['Single', 'Together', 'Married', 'Divorced', 'Widow', 'Alone',
       'Absurd', 'YOLO'], dtype=object)
```

In [24]:
```
#check to see how many inputted those odd entries
df.Marital_Status.value_counts()
```

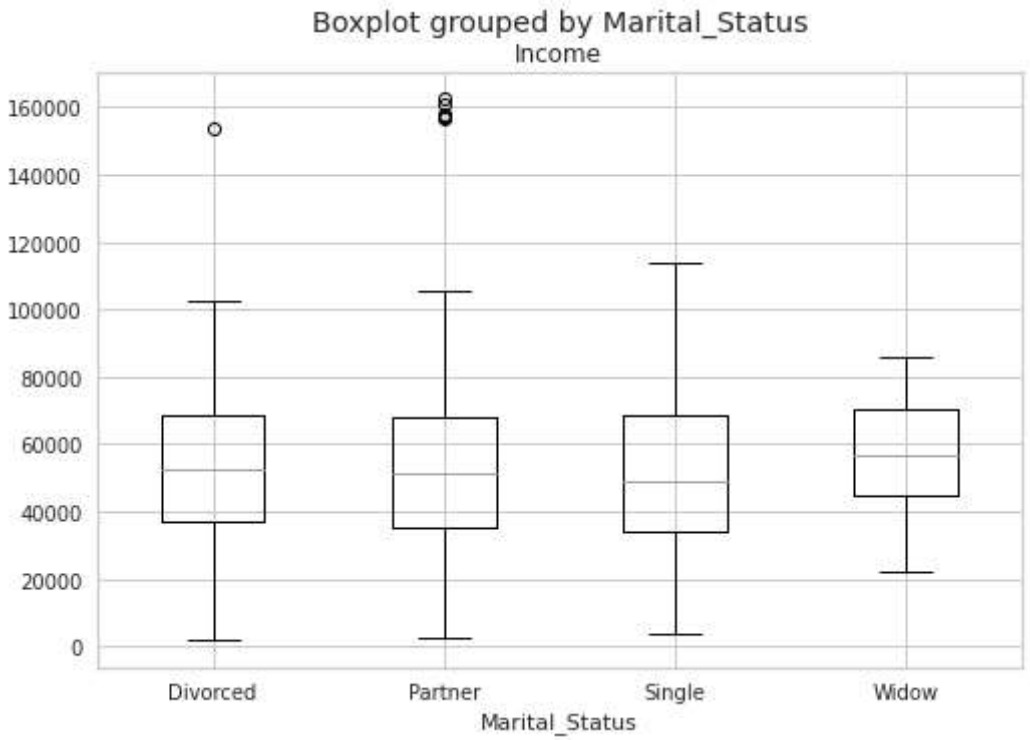
Out[24]:
```
Married     864
Together    579
Single      480
Divorced    232
Widow        77
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64
```

In [25]:
```
#Replaced alone with single, removed absurd and yolo, and combined married and
together into one category
df['Marital_Status']=df.Marital_Status.replace({'Alone':'Single'})
df = df.drop(df[df['Marital_Status'] == 'Absurd'].index)
df = df.drop(df[df['Marital_Status'] == 'YOLO'].index)
df['Marital_Status'] = df.Marital_Status.replace({'Married':'Partner', 'Togeth
er':'Partner'})
```

In [26]:
```
#created a column to check if someone is a parent and if they had a partner
df['isParent'] = df.children.apply(lambda x: 1 if x > 0 else 0)
df['withPartner'] = df.Marital_Status.apply(lambda x: 1 if x == 'Partner' else
0)

#checks for whole family size
df['famsize'] = df.children + df.withPartner + 1
```

In [27]:
```
#recheck now that we have removed the other categories
df.boxplot(column = 'Income', by = 'Marital_Status')
```

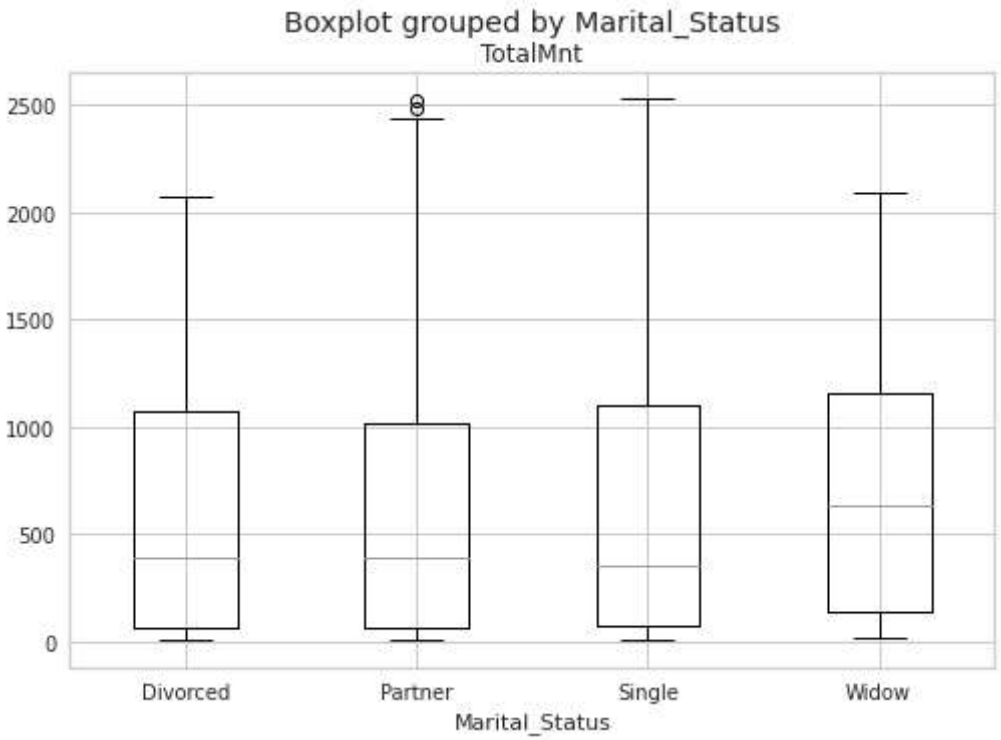
Out[27]: <AxesSubplot:title={'center':'Income'}, xlabel='Marital_Status'>



It does not seem like there is a difference in average income levels. There is a difference in range however, with widows having a smaller income range than the others.

In [28]:
```
df.boxplot(column = 'TotalMnt', by = 'Marital_Status')
```

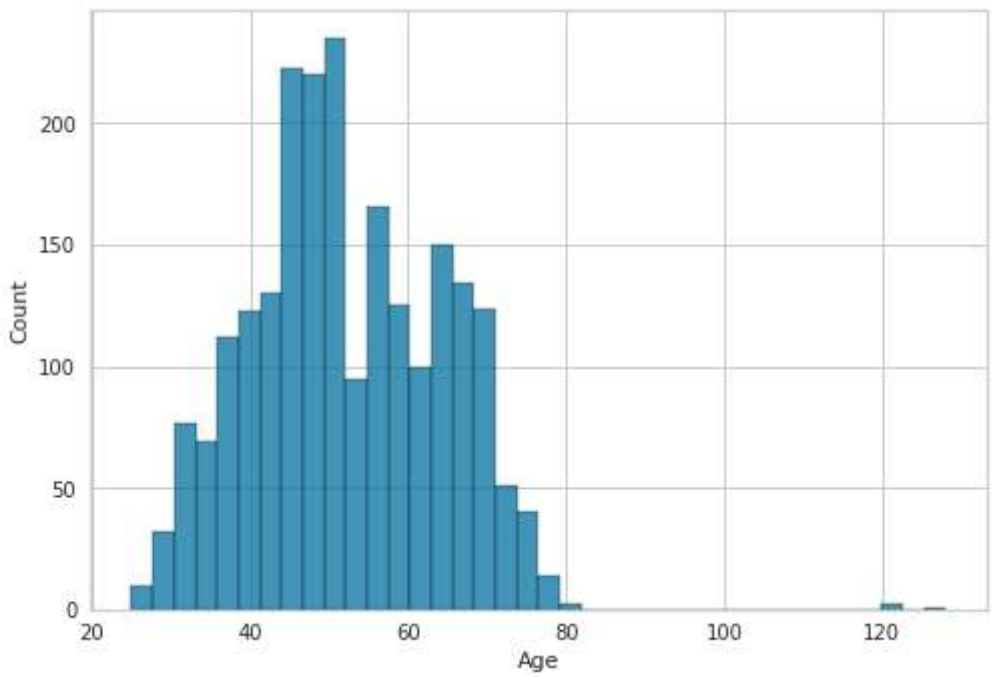
Out[28]:
```
<AxesSubplot:title={'center':'TotalMnt'}, xlabel='Marital_Status'>
```



For total amount spent, widows have the highest average expenditures, but those who have partners or are single, have the highest range.

In [29]:
```
#checked for the distribution of age and spot any potential outliers
sns.histplot(df.Age)
```

Out[29]: <AxesSubplot:xlabel='Age', ylabel='Count'>

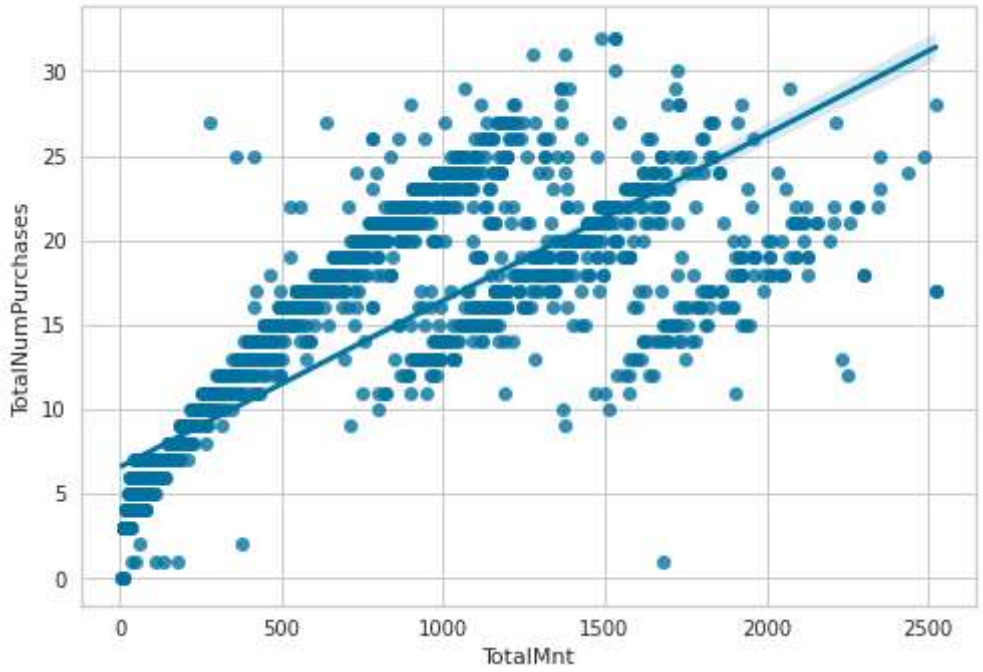


The distribution is relatively normal and is centered around 40 - 60 year olds. There is an outlier with some people who are older than 100.

In [30]:
```
#spotted an outlier and put an age cap to limit it
df = df.loc[df.Age < 100]
```

In [31]:
```
#check for the relationship between total amount spent and total number purhca
ses
sns.regplot(x = df.TotalMnt, y = df.TotalNumPurchases)
```

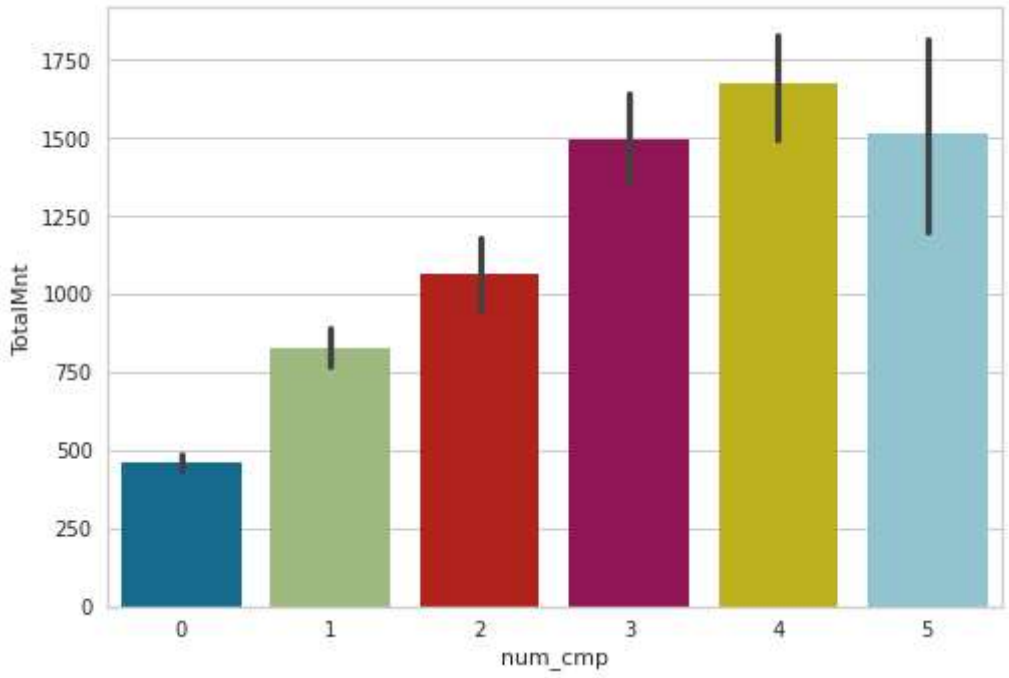
Out[31]: <AxesSubplot:xlabel='TotalMnt', ylabel='TotalNumPurchases'>



They seem to have a strong correlation with more purchases leading to higher sales.

In [32]:
```
#check to see if the number of successful campaigns increased sales
sns.barplot(y = 'TotalMnt', x = 'num_cmp', data = df)
```

Out[32]: <AxesSubplot:xlabel='num_cmp', ylabel='TotalMnt'>



It looks like more successful campaigns lead to more sales up until a certain point, then it started to decrease.

In [33]:
```
#drop redundant columns
fin_df=df.drop(["Year_Birth", "Z_CostContact", "Z_Revenue",'Kidhome','Teenhom
e'], axis=1)
df=df.drop(["Year_Birth", "Z_CostContact", "Z_Revenue",'Kidhome','Teenhome'],
axis=1)
fin_df
```

Out[33]:

	ID	Education	Marital_Status	Income	Dt_Customer	Recency	MntWines	MntFruits	M
0	5524	Graduation	Single	58138.0	2012-04-09	58	635	88	
1	2174	Graduation	Single	46344.0	2014-08-03	38	11	1	
2	4141	Graduation	Partner	71613.0	2013-08-21	26	426	49	
3	6182	Graduation	Partner	26646.0	2014-10-02	26	11	4	
4	5324	PhD	Partner	58293.0	2014-01-19	94	173	43	
...	...	...	...	...	...	...	...	...	
2235	10870	Graduation	Partner	61223.0	2013-06-13	46	709	43	
2236	4001	PhD	Partner	64014.0	2014-10-06	56	406	0	
2237	7270	Graduation	Divorced	56981.0	2014-01-25	91	908	48	
2238	8235	Master	Partner	69245.0	2014-01-24	8	428	30	
2239	9405	PhD	Partner	52869.0	2012-10-15	40	84	3	

2232 rows × 37 columns

After visualizing and creating the features it looks like we can now proceed to clustering the customers and we can begin by encoding some of our categorical data

In [34]:
```
#Encode categorical data
encode = LabelEncoder()
fin_df['Education']=encode.fit_transform(fin_df['Education'])
fin_df['Marital_Status']=encode.fit_transform(fin_df['Marital_Status'])
fin_df
```

Out[34]:

	ID	Education	Marital_Status	Income	Dt_Customer	Recency	MntWines	MntFruits	M
0	5524	2	2	58138.0	2012-04-09	58	635	88	
1	2174	2	2	46344.0	2014-08-03	38	11	1	
2	4141	2	1	71613.0	2013-08-21	26	426	49	
3	6182	2	1	26646.0	2014-10-02	26	11	4	
4	5324	4	1	58293.0	2014-01-19	94	173	43	
...	...	...	...	...	...	...	...	...	
2235	10870	2	1	61223.0	2013-06-13	46	709	43	
2236	4001	4	1	64014.0	2014-10-06	56	406	0	
2237	7270	2	0	56981.0	2014-01-25	91	908	48	
2238	8235	3	1	69245.0	2014-01-24	8	428	30	
2239	9405	4	1	52869.0	2012-10-15	40	84	3	

2232 rows × 37 columns

In [35]:
```
#Got an error about infinite values and I used this to check where the infinit
e values were
np.isinf(fin_df.iloc[:]).sum()
```

Out[35]:
```
ID                   0
Education            0
Marital_Status       0
Income               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
Response             0
Age                  0
TotalMnt             0
children             0
Income_to_spend      0
TotalNumPurchases    0
num_discounted       2
web_to_total         0
catalog_to_total     0
Store_to_total       0
num_cmp              0
isParent             0
withPartner          0
famsize              0
dtype: int64
```

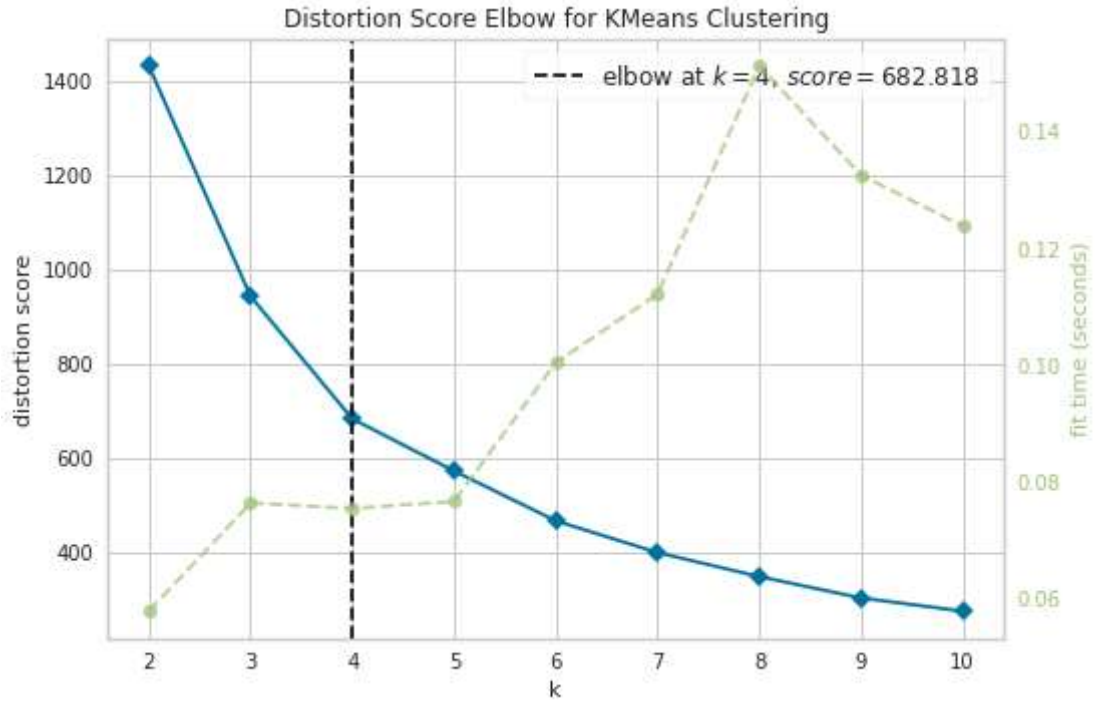
In [36]:
```
#replace infinite values with null and then drop the null rows
fin_df.replace([np.inf, -np.inf], np.nan, inplace=True)
fin_df.dropna(axis = 0,inplace=True)
df.replace([np.inf, -np.inf], np.nan, inplace=True)
df.dropna(axis = 0,inplace=True)
np.all(np.isfinite(fin_df))
```

Out[36]:
```
True
```

Now that our data has been encoded, we will now be clustering the customers by income and total amount spent. I decided to cluster using these features because it will allow us to clearly separate our customers depending on how much they earn and how much they spend. We can then analyze the other features in relation to those clusters to understand the characteristics of those that spend a lot, in order to come up with strategies that can increase the number of customers.

In [37]:
```
#scale the values for kmeans clustering
features = ['Income', 'TotalMnt']
fin_df2 = fin_df[features]
fin_df2 = (fin_df2 - fin_df2.mean(axis=0)) / fin_df2.std(axis=0)
#fin_df2 = (fin_df - fin_df.mean(axis=0)) / fin_df.std(axis=0)
```

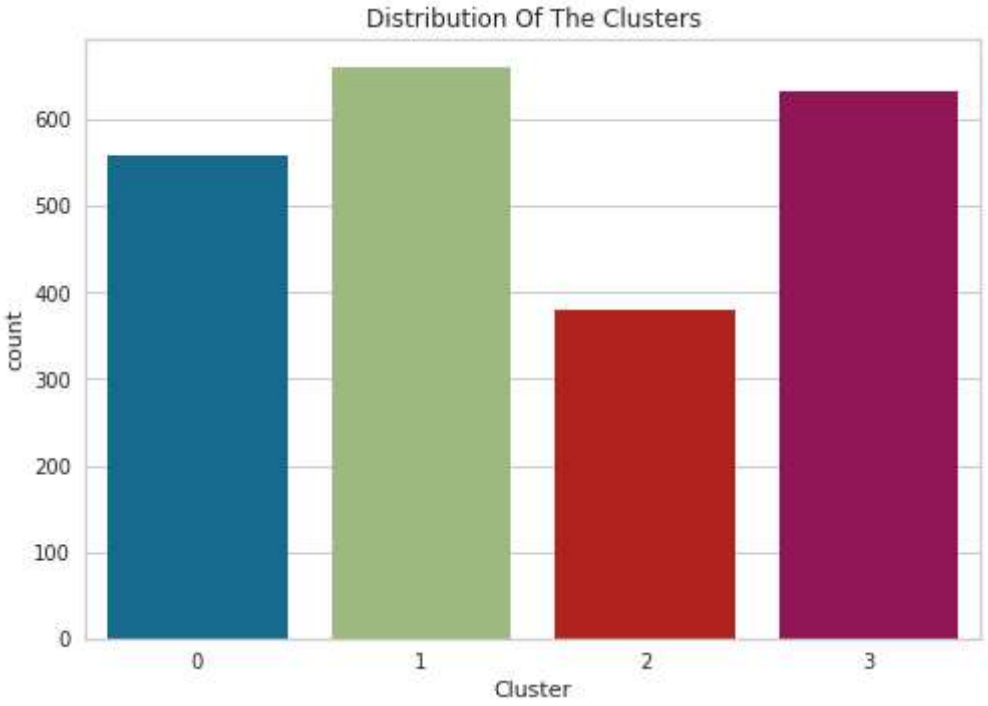
In [38]:
```
#check the optimal amount of clusters
Elbow_M = KElbowVisualizer(KMeans(), k=10)
Elbow_M.fit(fin_df2)
Elbow_M.show()
```



Out[38]:
```
<AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'},
xlabel='k', ylabel='distortion score'>
```

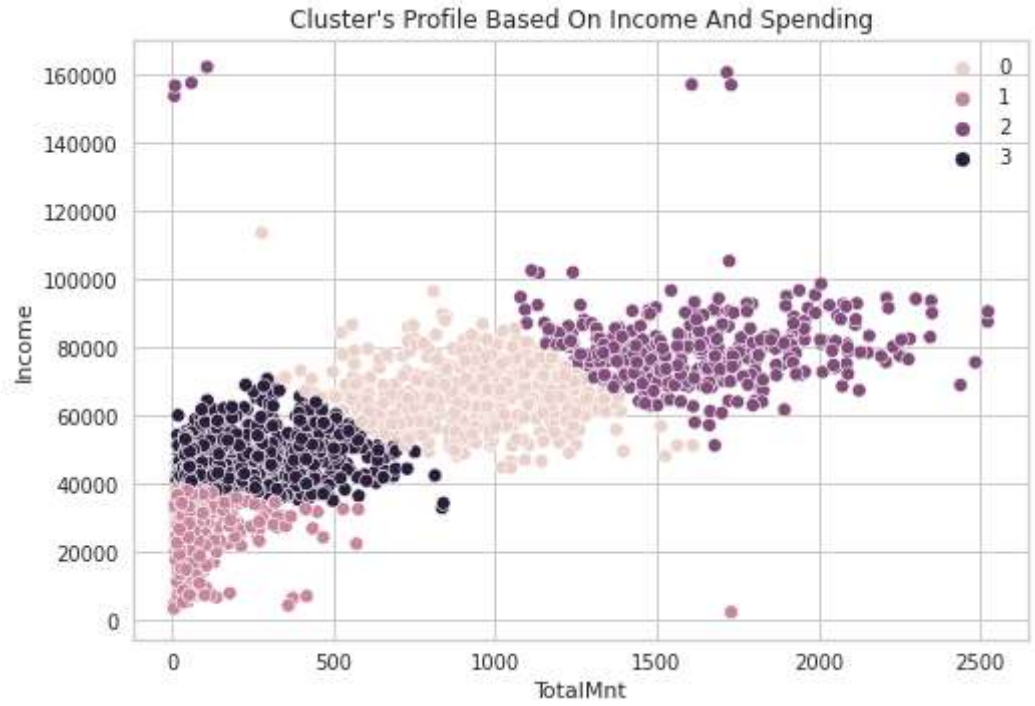
In [39]:
```
#cluster using kmeans
kmeans = KMeans(n_clusters=4, random_state=0)
fin_df["Cluster"] = kmeans.fit_predict(fin_df2)
df['Cluster']=fin_df['Cluster']
```

In [40]:
```
#chcek the distribution of the clusters
pl = sns.countplot(x=fin_df["Cluster"])
pl.set_title("Distribution Of The Clusters")
plt.show()
```



The distribution seems to be fairly even

In [41]:
```
#check if there is a clear distinction between clusters with reference to inco
me and total amount spent
pl = sns.scatterplot(data = fin_df,x=fin_df["TotalMnt"], y=fin_df["Income"],hu
e=fin_df["Cluster"])
pl.set_title("Cluster's Profile Based On Income And Spending")
plt.legend()
plt.show()
```



We can see here that we have 4 distinct clusters that describe the spending habits of the customers.

cluster 0 - high income, medium expenditure

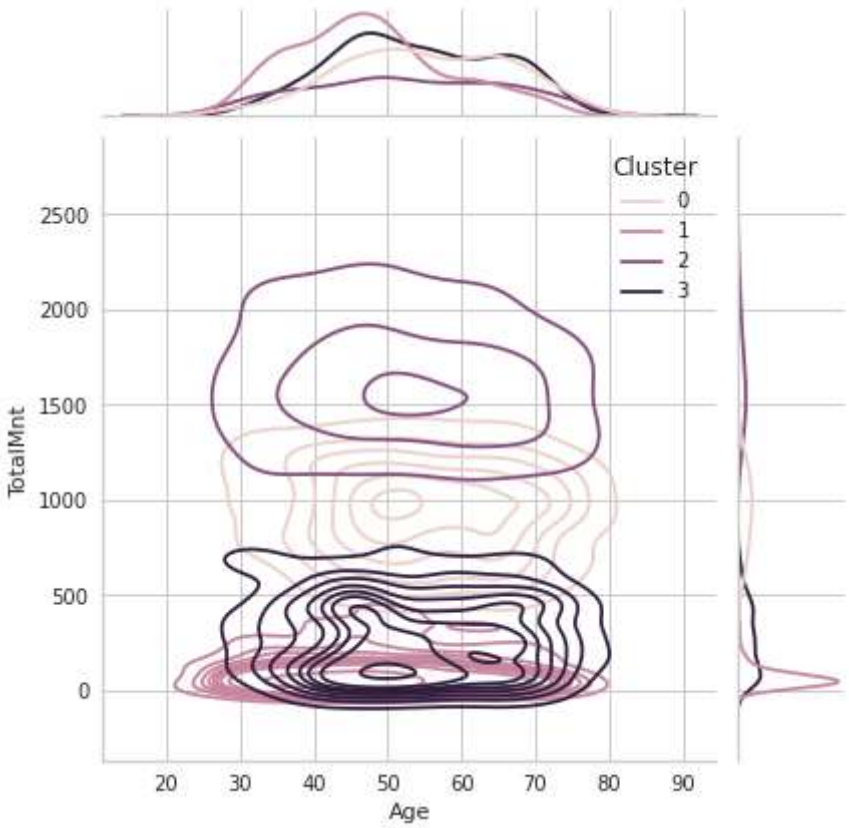
cluster 1 - low income, low expenditure

cluster 2 - high income, high expenditure

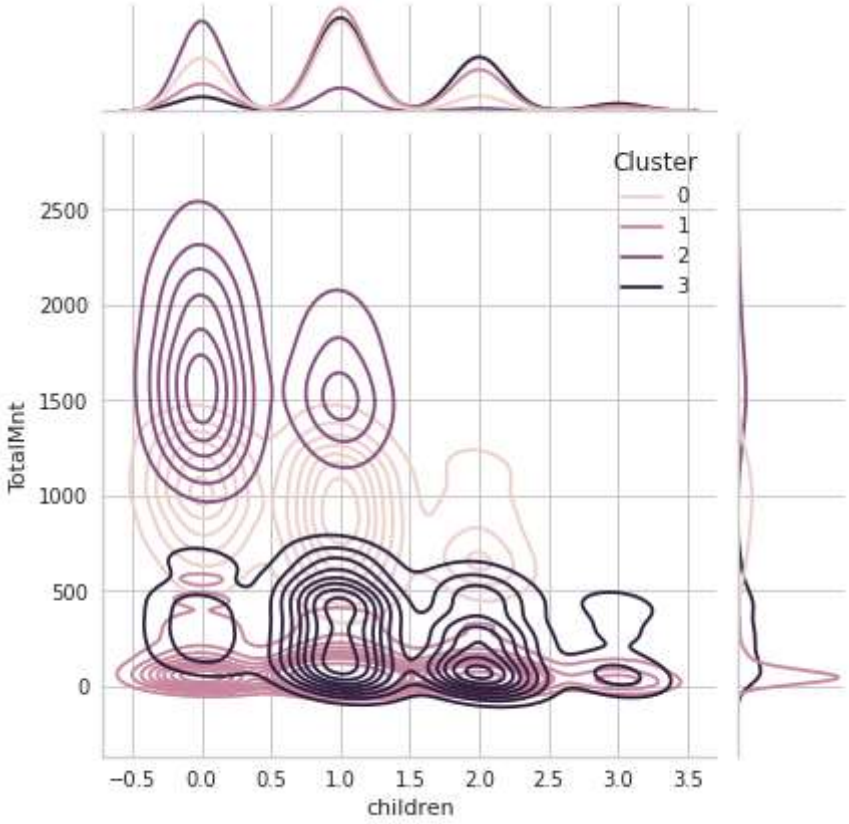
cluster 3 - medium income, low expenditure

In [42]:
```
#identify the personal attributes of the clusters
Personal = ["Age", "children", "famsize", "isParent", "withPartner"]

for i in Personal:
    plt.figure()
    sns.jointplot(x=fin_df[i], y=fin_df["TotalMnt"], hue =fin_df["Cluster"], k
ind="kde")
    plt.show()
```

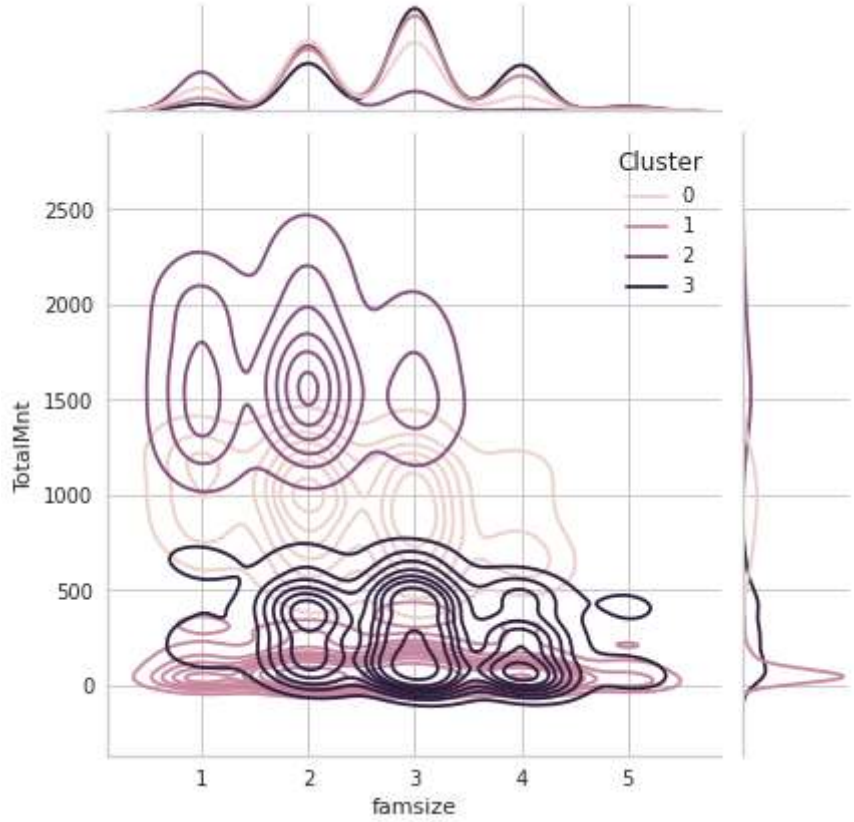
```
<Figure size 576x396 with 0 Axes>
```



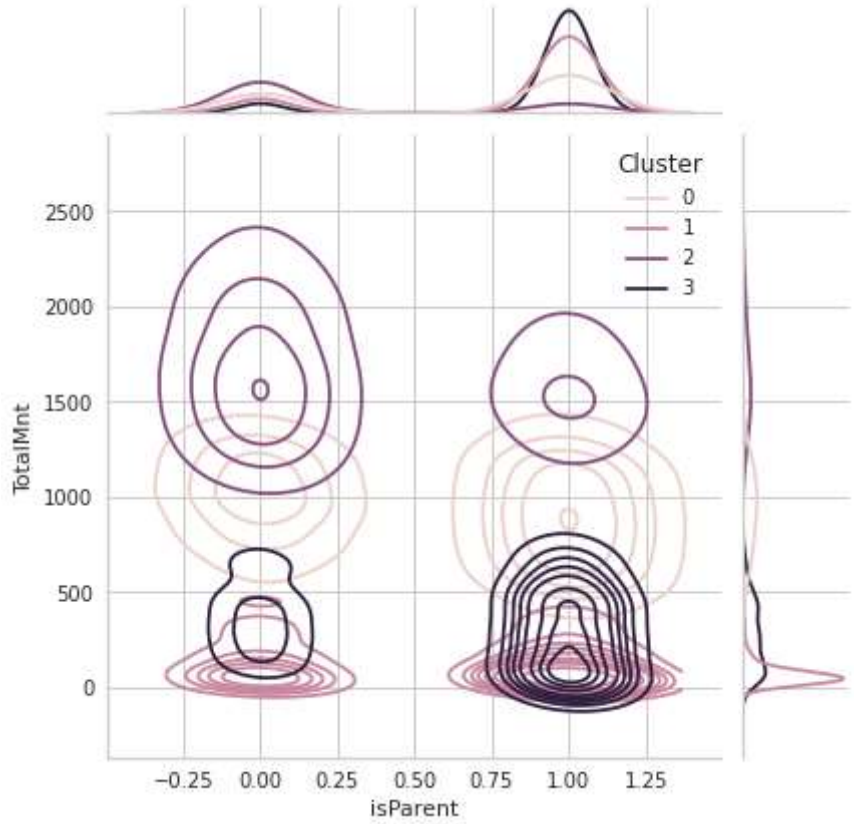

```
<Figure size 576x396 with 0 Axes>
```



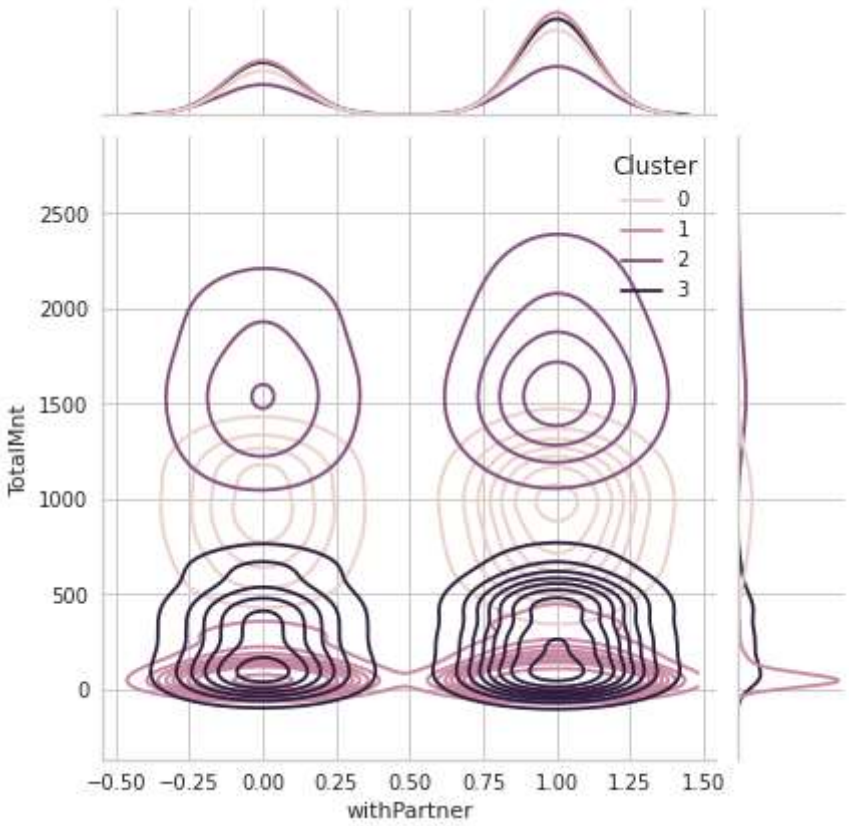

```
<Figure size 576x396 with 0 Axes>
```



<Figure size 576x396 with 0 Axes>

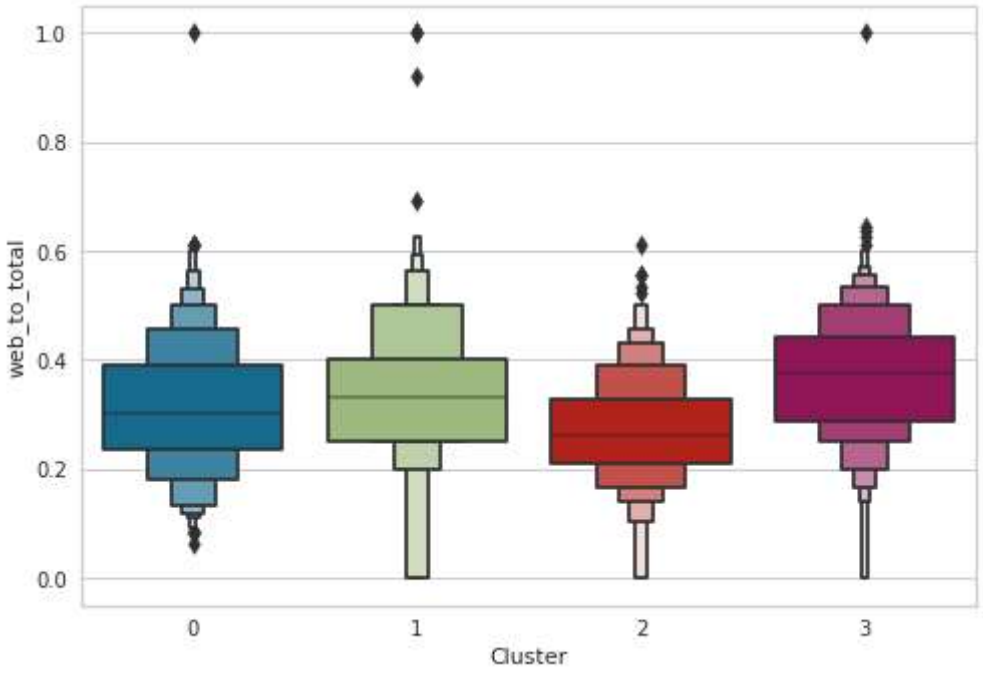


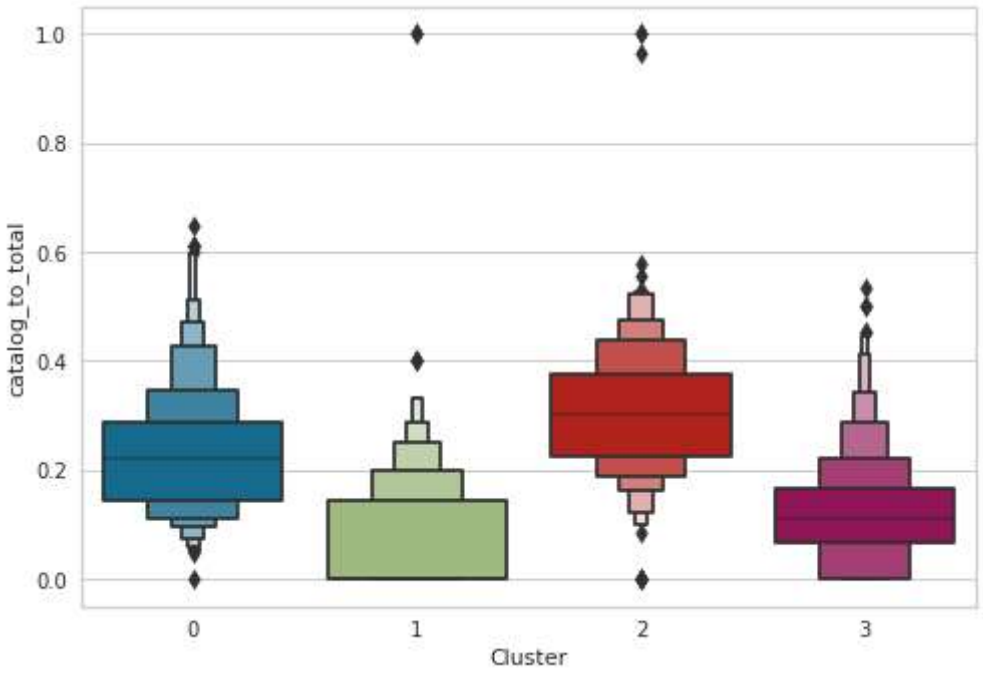
<Figure size 576x396 with 0 Axes>

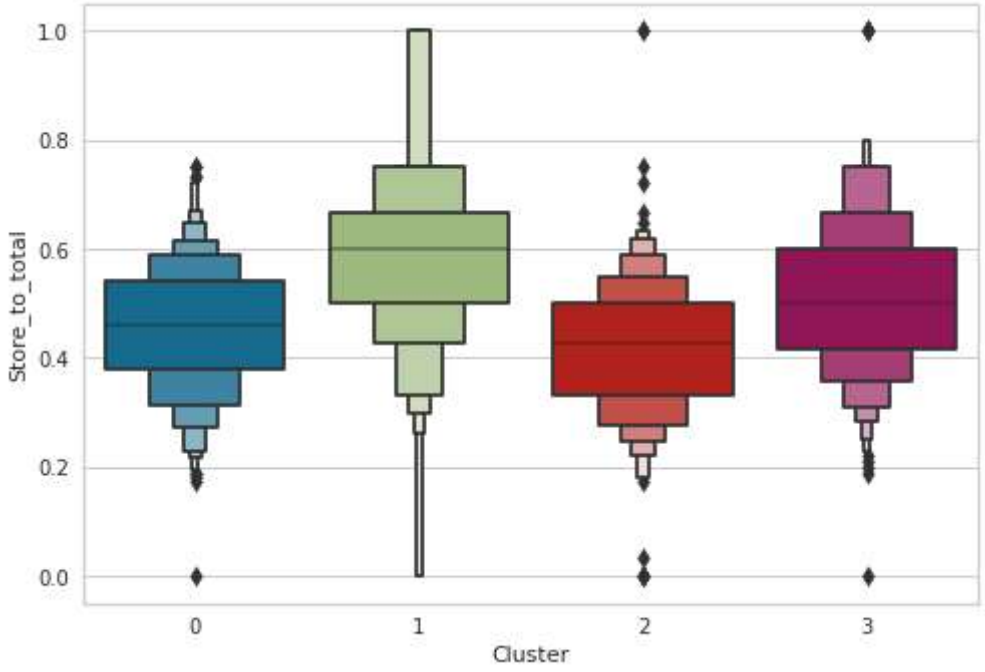


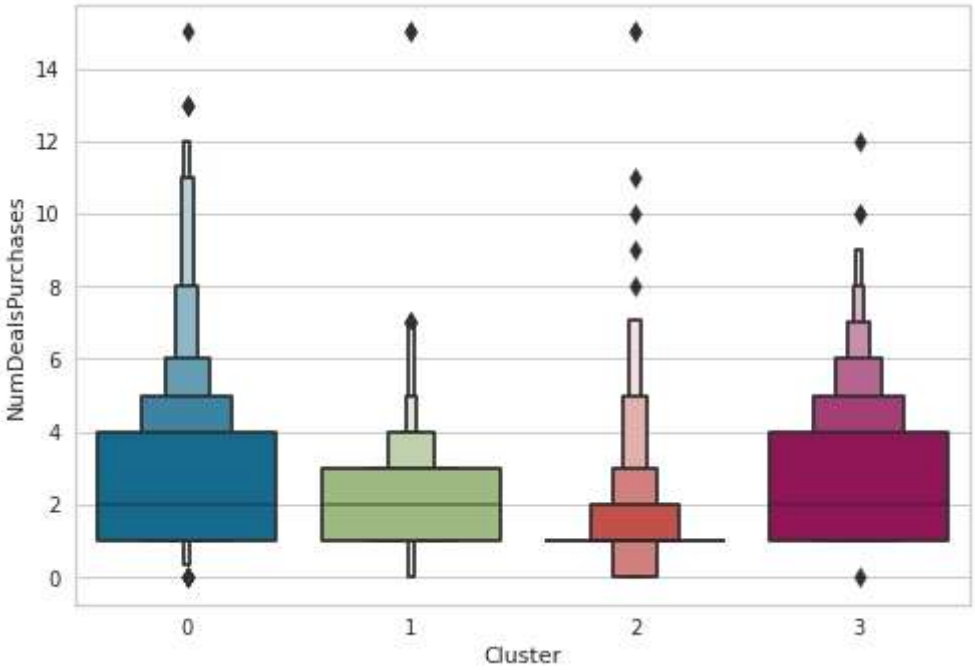
Based on these graphs, we can see that the ages seem to vary along with their marital status. The different clusters can have multiple children but those who spend more and earn more typically have less children.

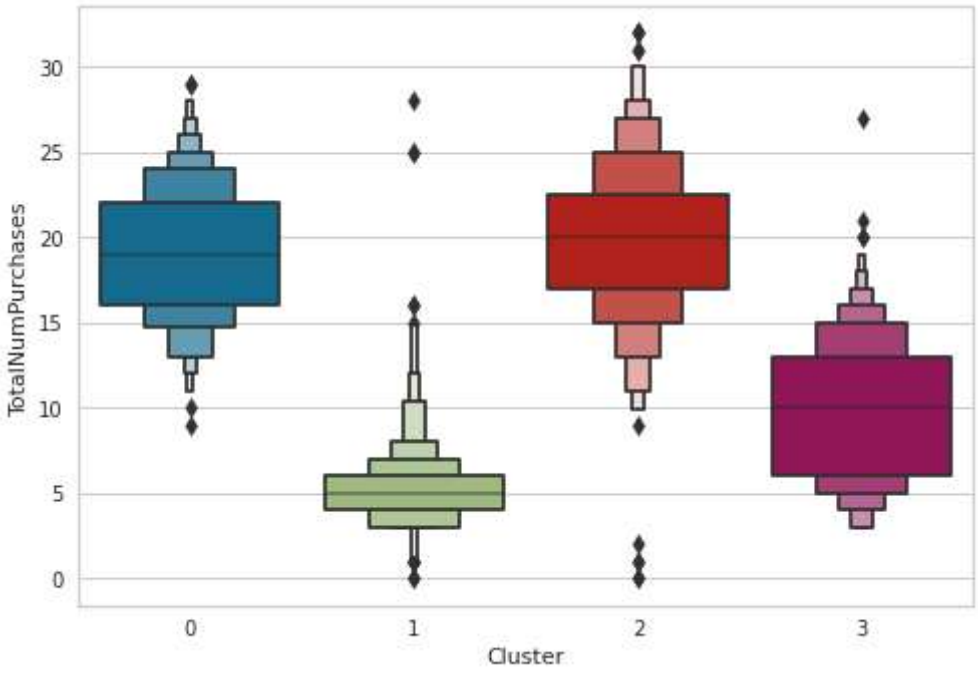
In [43]:
```
#Used ratios to better compare the mode of purchase between the clusters
trend = ["web_to_total", "catalog_to_total", "Store_to_total", "NumDealsPurcha
ses", "TotalNumPurchases"]

for i in trend:
    plt.figure()
    sns.boxenplot(y=fin_df[i], x =fin_df["Cluster"])
    plt.show()
```


web_to_total
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
Cluster


catalog_to_total
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
Cluster


Store_to_total
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
Cluster

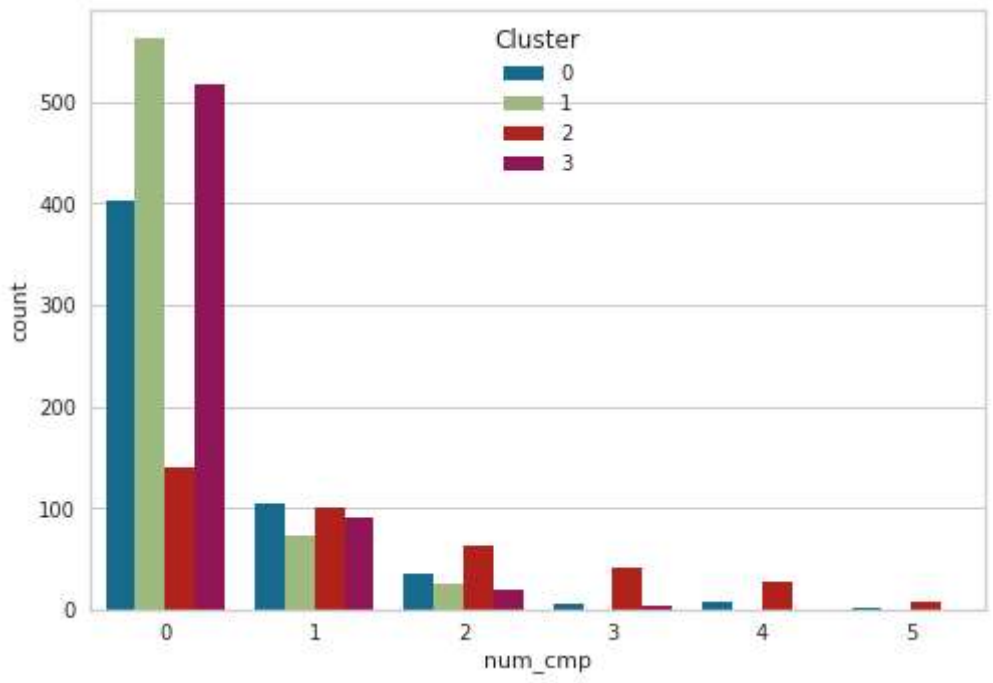

NumDealsPurchases
14
12
10
8
6
4
2
0
0
1
2
3
Cluster



These boxplot tell us the preferred method of purchase for the different clusters. We can see that the favored method of purchase for all clusters are through the store and those that prefer it more than the others are clusters 1 and 3. Second to that are web purchases which on average make up about 30% of the purchases for each cluster. The method that is least used is through catalog and clusters 1 and 3 purchase from catalogs significantly less so than the other 2 clusters. It seems most of the clusters take advantage of the deals, but cluster 0 and 3 are the one who are most likely to do so.

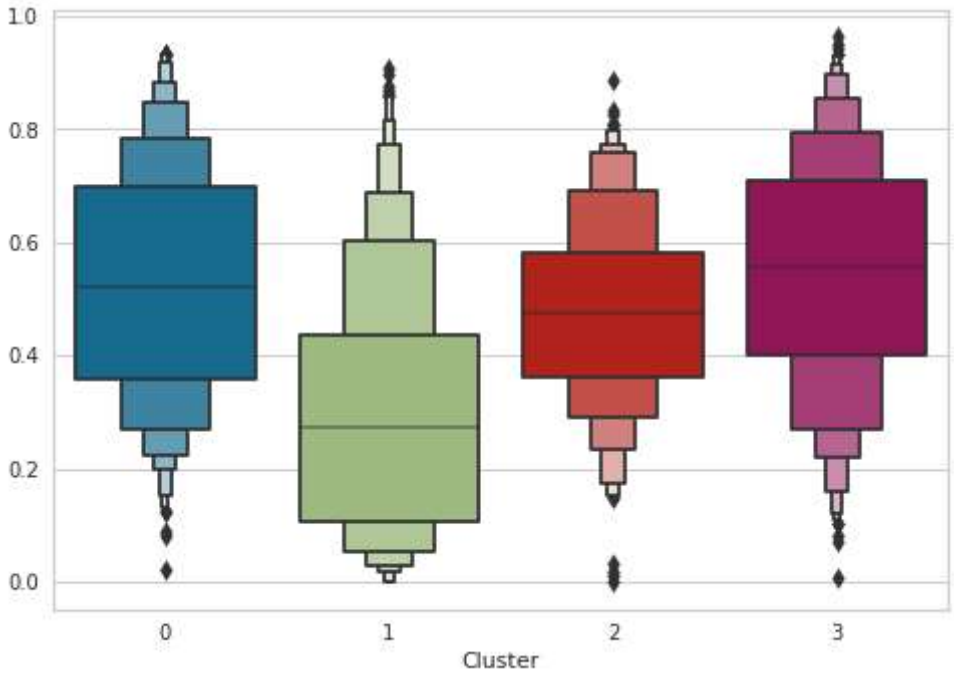
In [44]:
```
#check whether the campaigns were effective
sns.countplot(x=fin_df["num_cmp"],hue=fin_df["Cluster"])
```

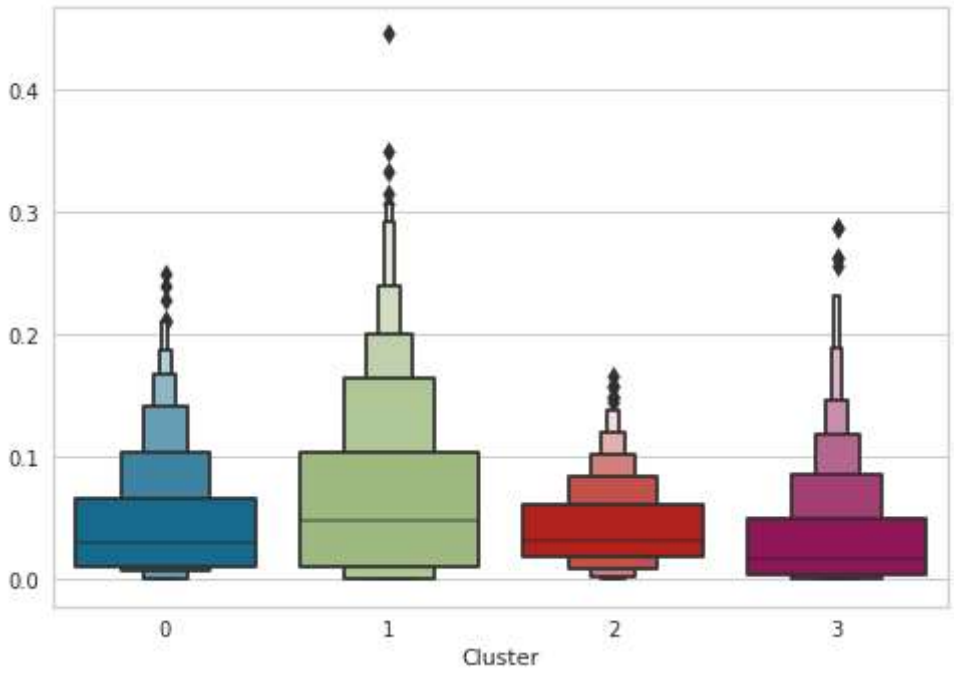
Out[44]:
```
<AxesSubplot:xlabel='num_cmp', ylabel='count'>
```

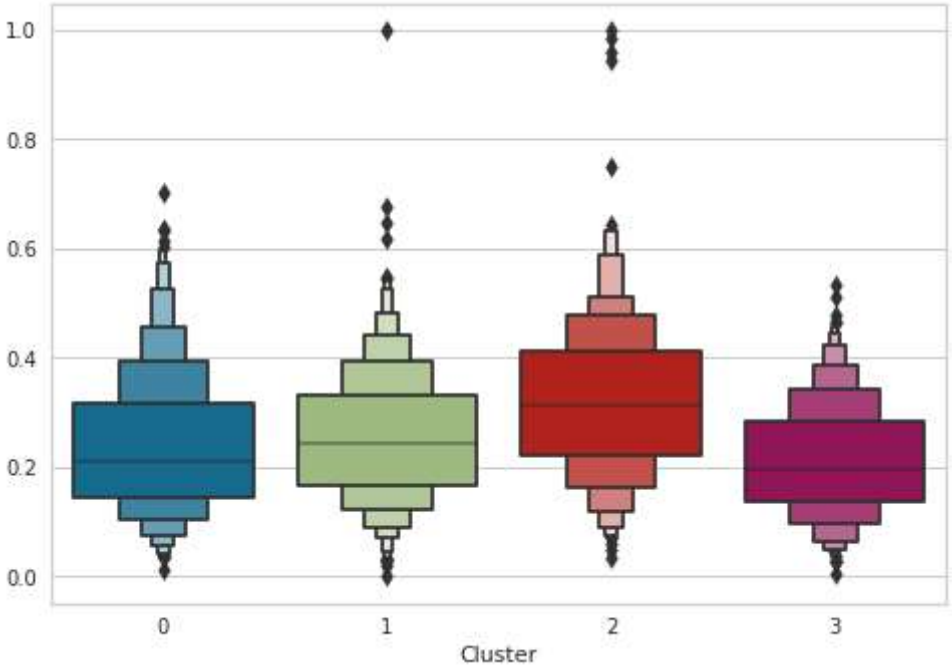


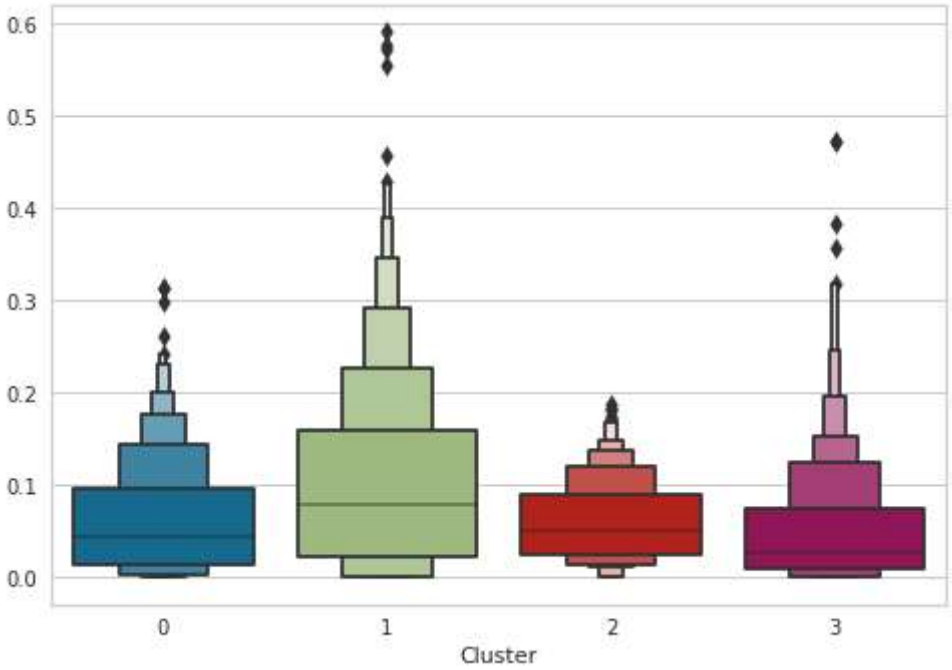
It does not seem like the campaign were particularly effective. Most of the clusters did not take part in any of the campaigns. The one exception to this is cluster 2 where it is a bit more spread out and most of the cluster has taken part in 1-3 of the campaigns

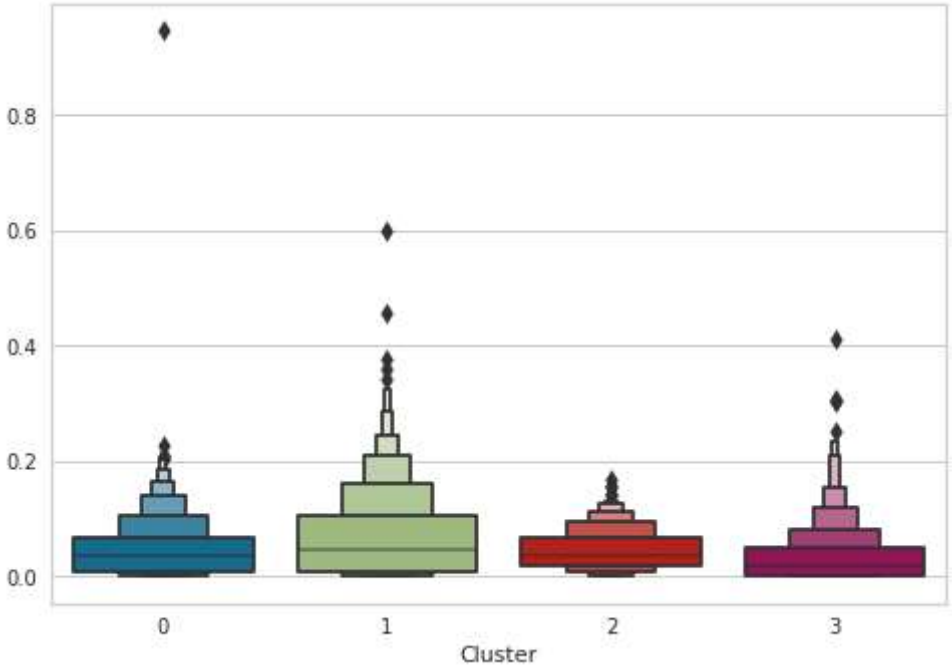
In [45]:
```
#check the amount to total ratio to be able to better compare product purchase
s between clusters
product = ["MntWines", "MntFruits", "MntMeatProducts", "MntFishProducts","MntS
weetProducts", "MntGoldProds"]

for i in product:
    plt.figure()
    sns.boxenplot(y=(fin_df[i]/fin_df['TotalMnt']),  x=fin_df["Cluster"])
    plt.show()
```

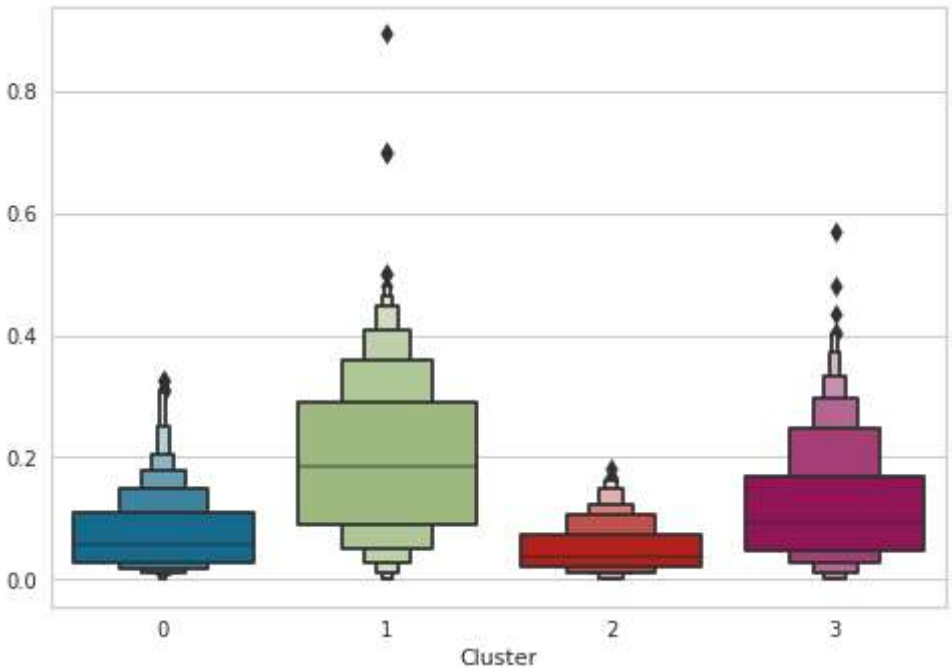

1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
Cluster


0.4
0.3
0.2
0.1
0.0
0
1
2
3
Cluster


1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
Cluster


0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
3
Cluster





It seems that most of the clusters spend most of their money of wine, the one exception to this is cluster 1 who spent the least amount on wine. The second product the customers spent the most on were meats and most of the clusters hover around a similar percent. Cluster 1 seems to have a wide range when purchasing fruits and fish. Cluster 1 along with cluster 3 are also the ones that spend the most on gold compared to the others.

cluster 0 - high income and medium expenses, varied ages, may or may not have kids, prefers stores, more liekly to take discounts

cluster 1 - low income and low expenses, varied ages, most likely has kids, mainly buys from stores even more than the other clusters, does not buy from catalogs much

cluster 2 - high income and high expenses, varied ages, most likely does not have kids, prefers stores, more likely to take part in campaigns

cluster 3 - medium income and low expenses, varied ages, most likely has 1-2 children, does not buy from catalogs much, prefers stores, more likely to buy discounted products

In [46]:
```
df.to_csv('/kaggle/working/df.csv')
```