Mini Project 3

Problem Statement

Customer Analysis is a detailed analysis of a company's customers. It helps a business to better understand its customers and makes it easier for them to modify products according to the specific needs, behaviours and concerns of different types of customers. Customer analysis helps a business to modify its product based on its target customers from different types of customer segments. For example, instead of spending money to market a new product to every customer in the company's database, a company can analyze which customer segment is most likely to buy the product and then market the product only on that particular segment.

Data Dictionary

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

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

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

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

Perform clustering to summarize customer segments.

Importing Liberaries for Data Processing

In [1]:
```
import numpy as np
import pandas as pd
```

Importing Data Visualization Liberies

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Preprocessing

In [3]:
```
# preprocessing
from sklearn.preprocessing import StandardScaler
```

Importing PCA

In [4]:
```
# pca
from sklearn.decomposition import PCA
```

Importing Unsupervised Machine Learning Liberaries

In [5]:
```
# clustering
from apyori import apriori
from sklearn.cluster import KMeans, AgglomerativeClustering
```

Importing CM For Evaluating

In [6]:
```
# evaluations
from sklearn.metrics import confusion_matrix
```

Importing Clusturing Liberaries

In [7]:
```
!pip install yellowbrick
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: yellowbrick in c:\users\admin\appdata\roaming\python\python39\site-packages (1.5)
Requirement already satisfied: scikit-learn>=1.0.0 in d:\anaconda3\lib\site-packages (from yellowbrick) (1.0.2)
Requirement already satisfied: cycler>=0.10.0 in d:\anaconda3\lib\site-packages (from yellowbrick) (0.11.0)
Requirement already satisfied: scipy>=1.0.0 in d:\anaconda3\lib\site-packages (from yellowbrick) (1.9.1)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in d:\anaconda3\lib\site-packages (from yellowbrick) (3.5.2)
Requirement already satisfied: numpy>=1.16.0 in d:\anaconda3\lib\site-packages (from yellowbrick) (1.21.5)
Requirement already satisfied: python-dateutil>=2.7 in d:\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in d:\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (9.2.0)
Requirement already satisfied: pyparsing>=2.2.1 in d:\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (3.0.9)
Requirement already satisfied: packaging>=20.0 in d:\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in d:\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in d:\anaconda3\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.4.2)
Requirement already satisfied: joblib>=0.11 in c:\users\admin\appdata\roaming\python\python39\site-packages (from scikit-learn>=1.0.0->yellowbrick) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\anaconda3\lib\site-packages (from scikit-learn>=1.0.0->yellowbrick) (2.2.0)
Requirement already satisfied: six>=1.5 in d:\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.16.0)
```

In [8]:
```
# clustering
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans, AgglomerativeClustering
```

Loading the Data

In [9]:
```
df=pd.read_csv("customer_data.csv")
```

Exploratory Data Analysis (EDA)

In [10]:
```
df.head()
```

Out[10]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWeb
0	5524	1957	Graduation	Single	58138.0	0	0	04/09/12	58	635	...	
1	2174	1954	Graduation	Single	46344.0	1	1	08/03/14	38	11	...	
2	4141	1965	Graduation	Together	71613.0	0	0	21/08/13	26	426	...	
3	6182	1984	Graduation	Together	26646.0	1	0	10/02/14	26	11	...	
4	5324	1981	PhD	Married	58293.0	1	0	19/01/14	94	173	...	

5 rows × 29 columns

In [11]:
```
df.tail()
```

Out[11]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	Num
2235	10870	1967	Graduation	Married	61223.0	0	1	13/06/13	46	709	...	
2236	4001	1946	PhD	Together	64014.0	2	1	10/06/14	56	406	...	
2237	7270	1981	Graduation	Divorced	56981.0	0	0	25/01/14	91	908	...	
2238	8235	1956	Master	Together	69245.0	0	1	24/01/14	8	428	...	
2239	9405	1954	PhD	Married	52869.0	1	1	15/10/12	40	84	...	

5 rows × 29 columns

In [12]:
```
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
```

In [13]:
```
df.shape
```

Out[13]: (2240, 29)

In [14]:
```
df.size
```

Out[14]: 64960

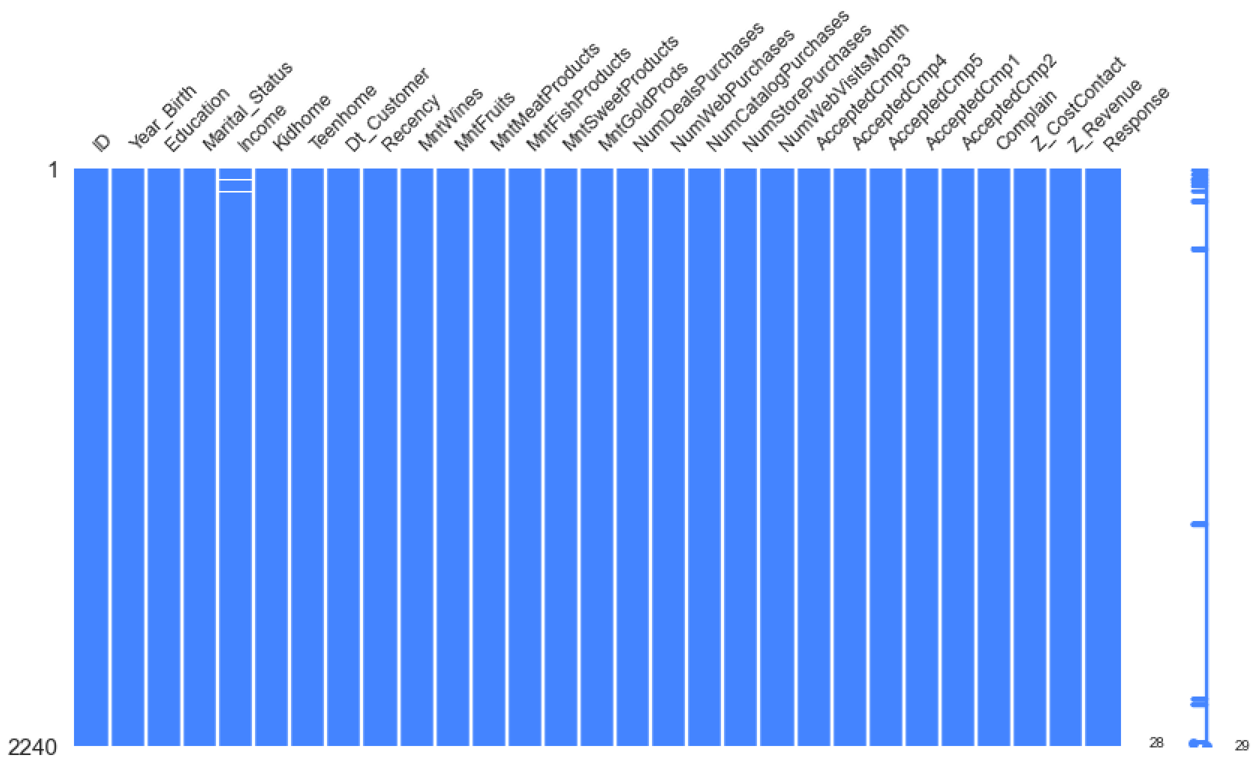
Treating Missing Values

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
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 [16]:
```
import missingno as msno # it will provides a small toolset of flexible and easy-to-use missing data v

msno.matrix(df, figsize=(10,5), fontsize=9,color=(0.27, 0.52, 1.0))
```

Out[16]: <AxesSubplot:>



In [17]:
```
df = df.dropna()
```

In [18]:
```
df.isnull().sum()
```

Out[18]:
```
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

In [19]:
```
df.describe()
```

Out[19]:

	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatF
count	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216
mean	5588.353339	1968.820397	52247.251354	0.441787	0.505415	49.012635	305.091606	26.356047	166
std	3249.376275	11.985554	25173.076661	0.536896	0.544181	28.948352	337.327920	39.793917	224
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	(
25%	2814.750000	1959.000000	35303.000000	0.000000	0.000000	24.000000	24.000000	2.000000	16
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000	49.000000	174.500000	8.000000	68
75%	8421.750000	1977.000000	68522.000000	1.000000	1.000000	74.000000	505.000000	33.000000	232
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725

8 rows × 26 columns

In [20]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2216 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2216 non-null   int64
 1   Year_Birth           2216 non-null   int64
 2   Education            2216 non-null   object
 3   Marital_Status       2216 non-null   object
 4   Income               2216 non-null   float64
 5   Kidhome              2216 non-null   int64
 6   Teenhome             2216 non-null   int64
 7   Dt_Customer          2216 non-null   object
 8   Recency              2216 non-null   int64
 9   MntWines             2216 non-null   int64
 10  MntFruits            2216 non-null   int64
 11  MntMeatProducts      2216 non-null   int64
 12  MntFishProducts      2216 non-null   int64
 13  MntSweetProducts     2216 non-null   int64
 14  MntGoldProds         2216 non-null   int64
 15  NumDealsPurchases    2216 non-null   int64
 16  NumWebPurchases      2216 non-null   int64
 17  NumCatalogPurchases  2216 non-null   int64
 18  NumStorePurchases    2216 non-null   int64
 19  NumWebVisitsMonth    2216 non-null   int64
 20  AcceptedCmp3         2216 non-null   int64
 21  AcceptedCmp4         2216 non-null   int64
 22  AcceptedCmp5         2216 non-null   int64
 23  AcceptedCmp1         2216 non-null   int64
 24  AcceptedCmp2         2216 non-null   int64
 25  Complain             2216 non-null   int64
 26  Z_CostContact        2216 non-null   int64
 27  Z_Revenue            2216 non-null   int64
 28  Response             2216 non-null   int64
dtypes: float64(1), int64(25), object(3)
memory usage: 519.4+ KB
```

In [21]:
```
# dt Customer is showing here is object where as these are date;
```

In [22]:
```
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'])
```

In [23]:
```
print("The Newest customer's entry date in the records : ", max(df['Dt_Customer']))
print("The Oldest customer's entry date in the records : ", min(df['Dt_Customer']))
```

```
The Newest customer's entry date in the records :  2014-12-06 00:00:00
The Oldest customer's entry date in the records :  2012-01-08 00:00:00
```

Finding Out "Age" of a customer by the "Year_Birth" indicating the birth year of the respective person.

In [24]:
```
df['Age'] = 2015 - df['Year_Birth']
```

Create another feature "Spent" signifies the total amount spent by the customer in various categories over the time of two years.

In [25]:
```
df['Total Spent'] = df['MntWines'] + df['MntFruits'] + df['MntMeatProducts'] + df['MntFishProducts'] 
```

Create another feature "Living_With" out of "Marital_Status" to extract the living situation of couples.

In [26]:
```
df["Marital_Status"].value_counts()
```

Out[26]:
```
Married     857
Together    573
Single      471
Divorced    232
Widow        76
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64
```

In [27]:
```
df['Living_With'] = df['Marital_Status'].replace({'Married':'Partner', 'Together':'Partner', 'Absurd'
```

Create a feature "Children" to indicate total children in a household i.e, kids and teenagers.

In [28]:
```
df["Kidhome"].value_counts()
```

Out[28]:
```
0    1283
1     887
2      46
Name: Kidhome, dtype: int64
```

In [29]:
```
df["Teenhome"].value_counts()
```

Out[29]:
```
0    1147
1    1018
2      51
Name: Teenhome, dtype: int64
```

In [30]:
```
df['Children'] = df['Kidhome'] + df['Teenhome']
```

To get further clarity of household, Creating feature indicating "Family_Size"

In [31]:
```
df['Family_Size'] = df['Living_With'].replace({'Alone': 1, 'Partner':2}) + df['Children']
```

Create a feature "Is_Parent" to indicate parenthood status

In [32]:
```
df['Is_Parent'] = np.where(df.Children > 0, 1, 0)
```

Segregating education levels in three groups

In [33]:
```
df["Education"].value_counts()
```

Out[33]:
```
Graduation    1116
PhD            481
Master         365
2n Cycle       200
Basic           54
Name: Education, dtype: int64
```

In [34]:
```
df['Education'] = df['Education'].replace({'Basic':'Undergraduate', '2n Cycle':'Undergraduate', 'Grad
```

Checking Our New DataFrame

In [35]:
```
df.head()
```

Out[35]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	Compl
0	5524	1957	Graduate	Single	58138.0	0	0	2012-04-09	58	635	...	
1	2174	1954	Graduate	Single	46344.0	1	1	2014-08-03	38	11	...	
2	4141	1965	Graduate	Together	71613.0	0	0	2013-08-21	26	426	...	
3	6182	1984	Graduate	Together	26646.0	1	0	2014-10-02	26	11	...	
4	5324	1981	Postgraduate	Married	58293.0	1	0	2014-01-19	94	173	...	

5 rows × 35 columns

Dropping some of the Unused Features

In [36]:
```
getout = ['Marital_Status', 'Dt_Customer', 'Z_CostContact', 'Z_Revenue', 'Year_Birth', 'ID']
df = df.drop(getout, axis=1)
```

Checking Our New DataFrame

In [37]:
```
df.head()
```

Out[37]:

	Education	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetP
0	Graduate	58138.0	0	0	58	635	88	546	172	
1	Graduate	46344.0	1	1	38	11	1	6	2	
2	Graduate	71613.0	0	0	26	426	49	127	111	
3	Graduate	26646.0	1	0	26	11	4	20	10	
4	Postgraduate	58293.0	1	0	94	173	43	118	46	

5 rows × 29 columns

Data Visualization And Data Analysis

In [38]:
```
df.shape
```

Out[38]: (2216, 29)

In [39]:
```
df.size
```

Out[39]: 64264

In [40]:
```
df.describe()
```

Out[40]:

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts
count	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000
mean	52247.251354	0.441787	0.505415	49.012635	305.091606	26.356047	166.995939	37.637635
std	25173.076661	0.536896	0.544181	28.948352	337.327920	39.793917	224.283273	54.752082
min	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	35303.000000	0.000000	0.000000	24.000000	24.000000	2.000000	16.000000	3.000000
50%	51381.500000	0.000000	0.000000	49.000000	174.500000	8.000000	68.000000	12.000000
75%	68522.000000	1.000000	1.000000	74.000000	505.000000	33.000000	232.250000	50.000000
max	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000

8 rows × 27 columns

In [41]:
```
df.describe(include=object).T
```

Out[41]:

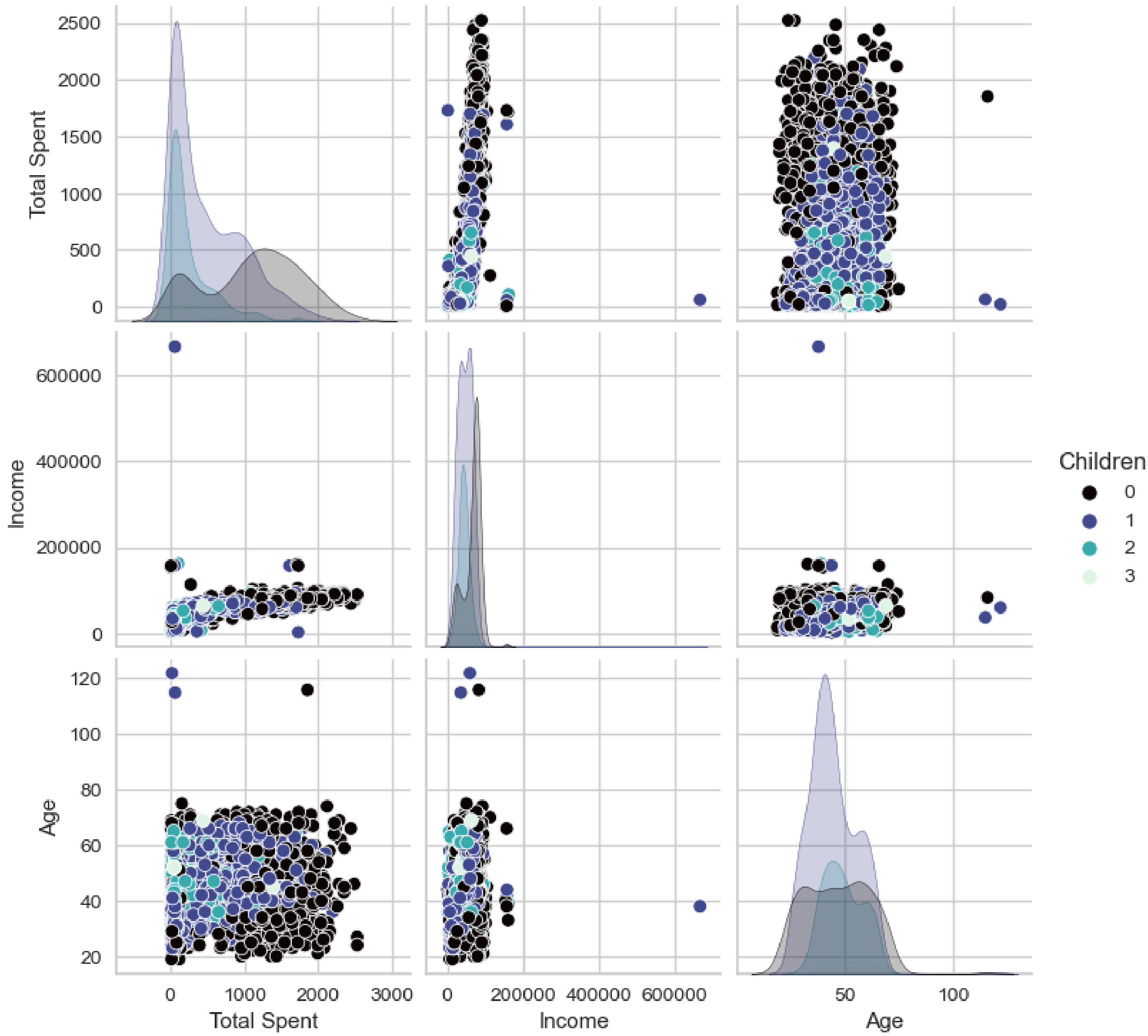
	count	unique	top	freq
Education	2216	3	Graduate	1116
Living_With	2216	2	Partner	1430

In [42]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2216 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Education            2216 non-null   object 
 1   Income               2216 non-null   float64
 2   Kidhome              2216 non-null   int64  
 3   Teenhome             2216 non-null   int64  
 4   Recency              2216 non-null   int64  
 5   MntWines             2216 non-null   int64  
 6   MntFruits            2216 non-null   int64  
 7   MntMeatProducts      2216 non-null   int64  
 8   MntFishProducts      2216 non-null   int64  
 9   MntSweetProducts     2216 non-null   int64  
 10  MntGoldProds         2216 non-null   int64  
 11  NumDealsPurchases    2216 non-null   int64  
 12  NumWebPurchases      2216 non-null   int64  
 13  NumCatalogPurchases  2216 non-null   int64  
 14  NumStorePurchases    2216 non-null   int64  
 15  NumWebVisitsMonth    2216 non-null   int64  
 16  AcceptedCmp3         2216 non-null   int64  
 17  AcceptedCmp4         2216 non-null   int64  
 18  AcceptedCmp5         2216 non-null   int64  
 19  AcceptedCmp1         2216 non-null   int64  
 20  AcceptedCmp2         2216 non-null   int64  
 21  Complain             2216 non-null   int64  
 22  Response             2216 non-null   int64  
 23  Age                  2216 non-null   int64  
 24  Total Spent          2216 non-null   int64  
 25  Living_With          2216 non-null   object 
 26  Children             2216 non-null   int64  
 27  Family_Size          2216 non-null   int64  
 28  Is_Parent            2216 non-null   int32  
dtypes: float64(1), int32(1), int64(25), object(2)
memory usage: 510.7+ KB
```

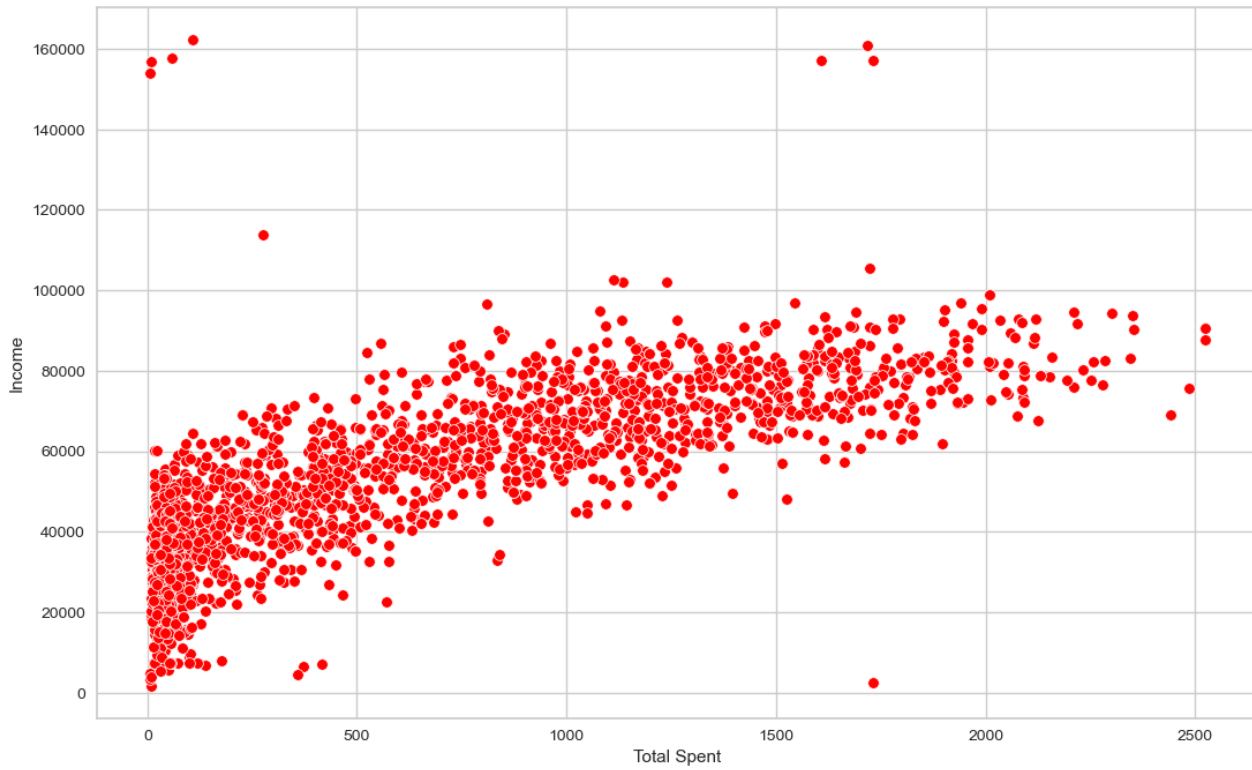
In [43]:
```
sns.pairplot(df , vars=['Total Spent','Income','Age'] , hue='Children', palette="mako")
```

Out[43]: <seaborn.axisgrid.PairGrid at 0x241b4964ac0>

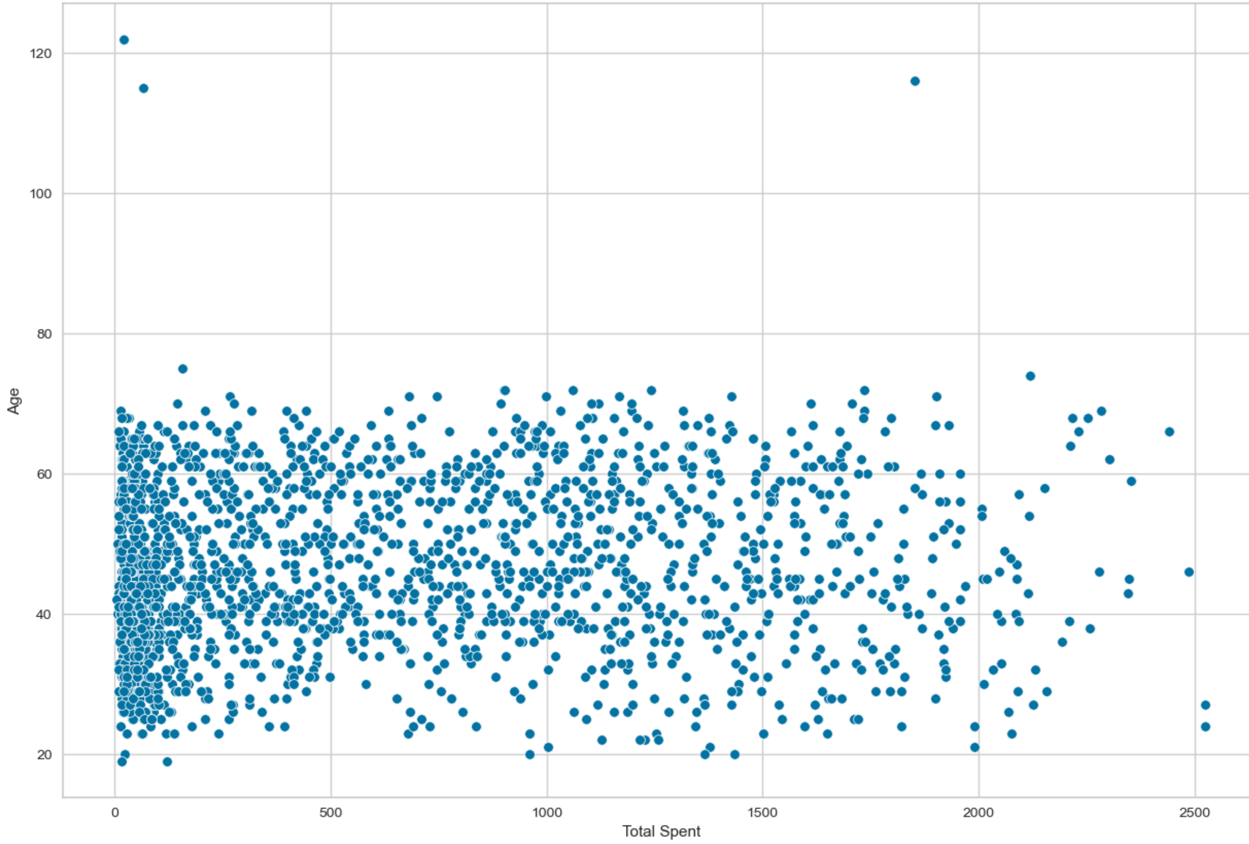


In [44]:
```
plt.figure(figsize=(13,8))
sns.scatterplot(x=df[df['Income']<600000]['Total Spent'], y=df[df['Income']<600000]['Income'], color=
```

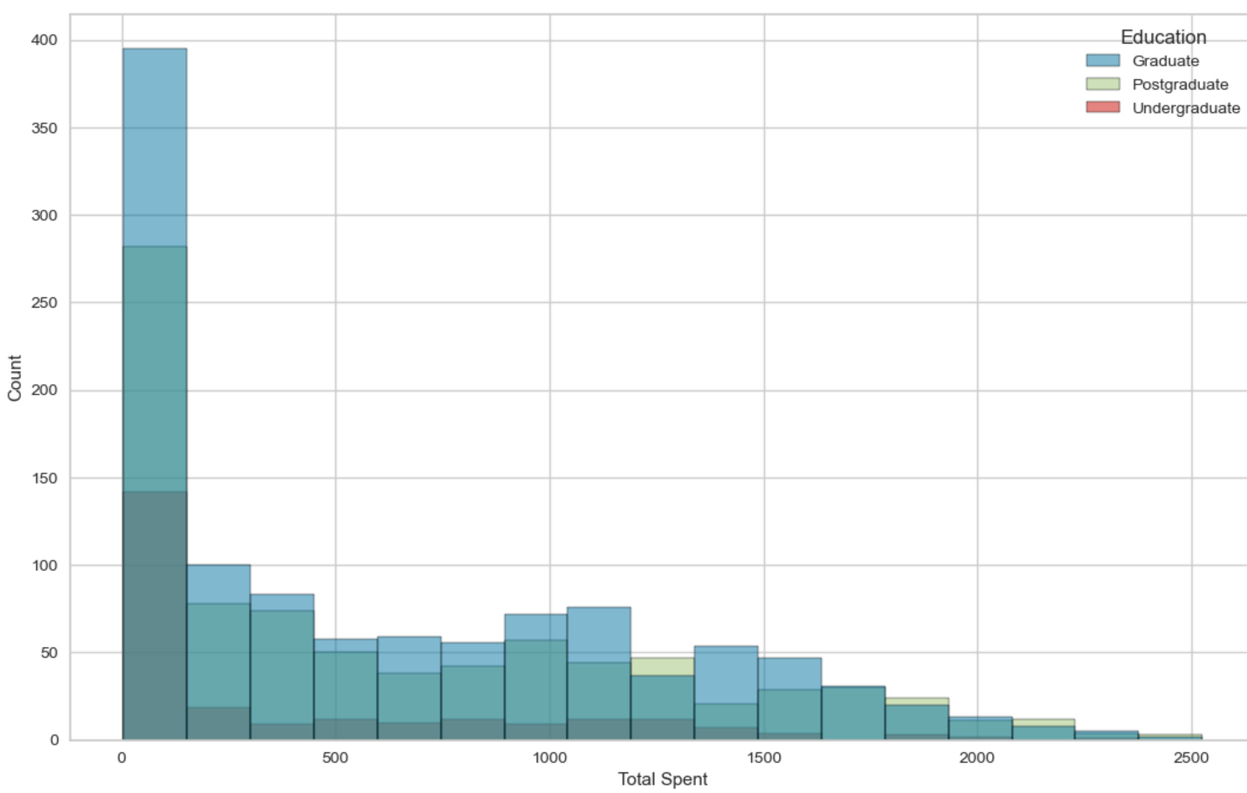
Out[44]: <AxesSubplot:xlabel='Total Spent', ylabel='Income'>



In [45]:
```
plt.figure(figsize=(15,10))
sns.scatterplot(x=df['Total Spent'], y=df['Age'])
plt.show()
```

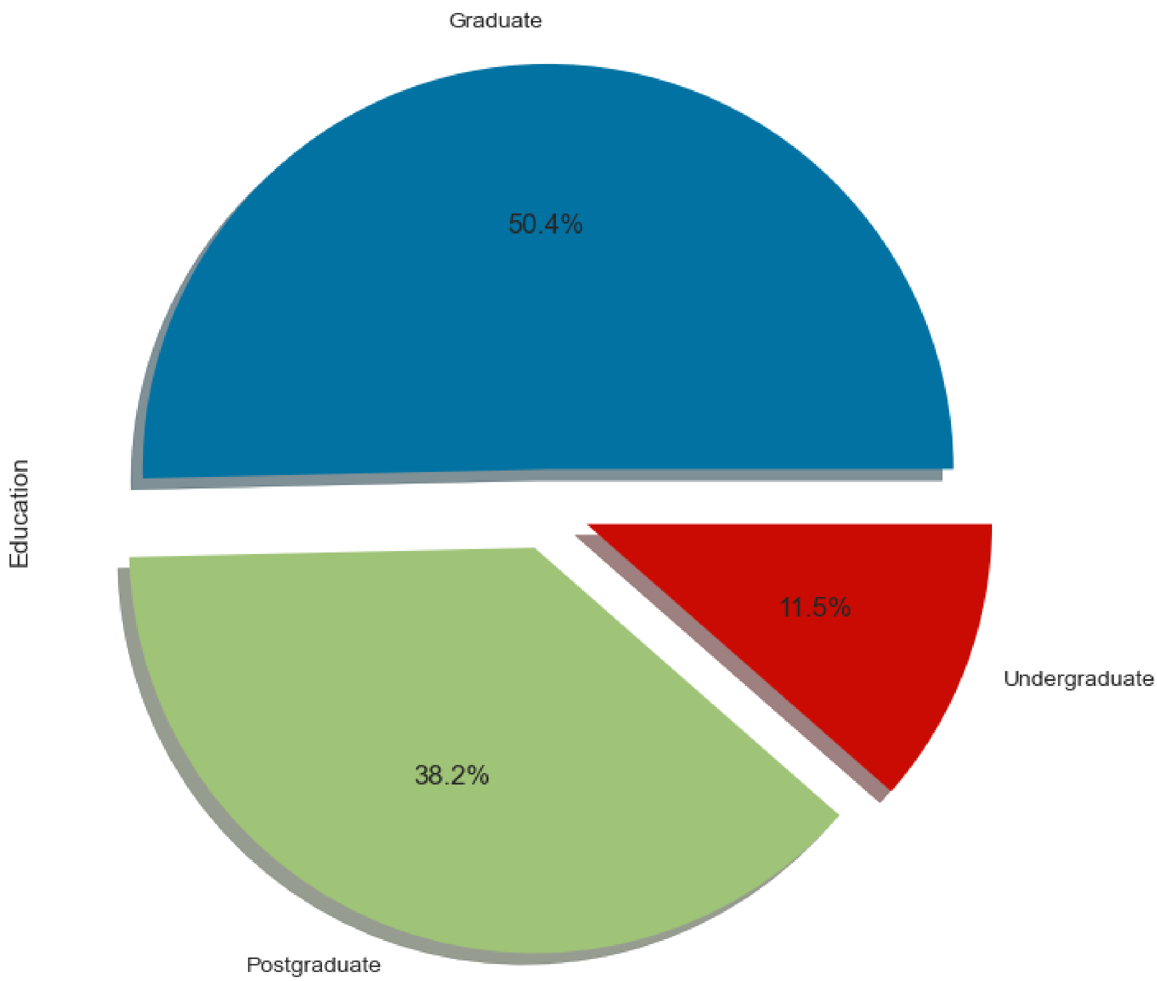


In [46]:
```
plt.figure(figsize=(13,8))
sns.histplot(x=df['Total Spent'], hue=df['Education'])
plt.show()
```



In [47]:
```
df['Education'].value_counts().plot.pie(explode=[0.1,0.1,0.1], autopct='%1.1f%%', shadow=True, figsiz
```

Out[47]:
```
<AxesSubplot:ylabel='Education'>
```

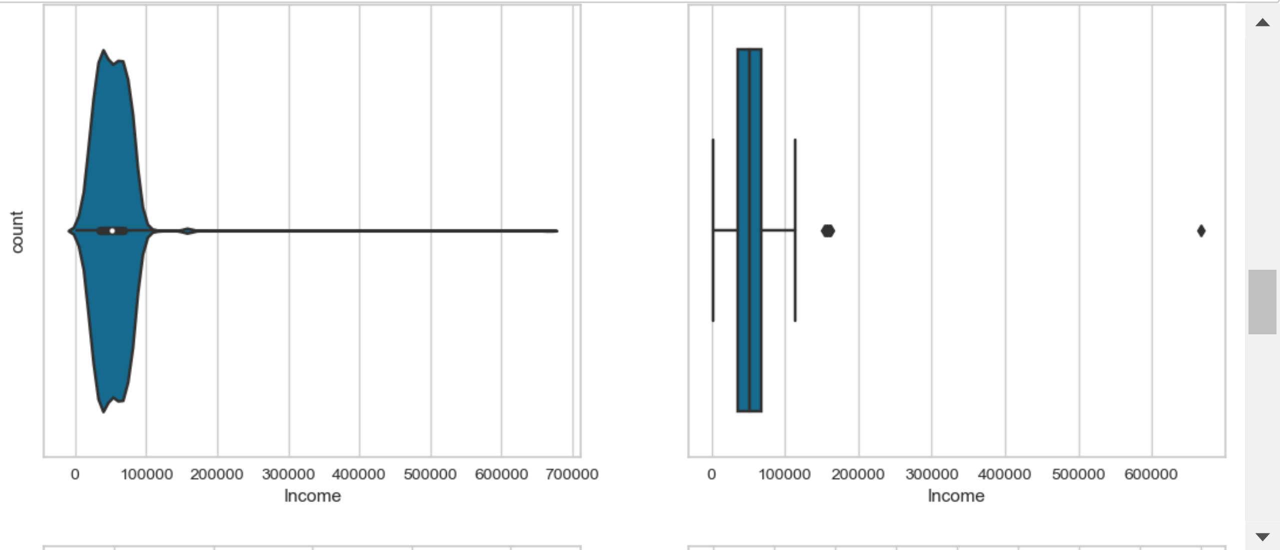


In [48]:
```
df["Education"].value_counts()
```

Out[48]:
```
Graduate        1116
Postgraduate     846
Undergraduate    254
Name: Education, dtype: int64
```

Outlier Detection

In [49]:
```
corr=df.corr()
fig, axes = plt.subplots(2,2, figsize=(20, 20))
for i, j in zip(corr[:29], axes.flatten()):
    print("Skewness : ", round(df[i].skew(),3))
    plt.figure(figsize=(13,5))
    plt.subplot(1,2,1)
    sns.violinplot(df[i])
    plt.ylabel('count')
    plt.subplot(1,2,2)
    sns.boxplot(x=df[i])
```



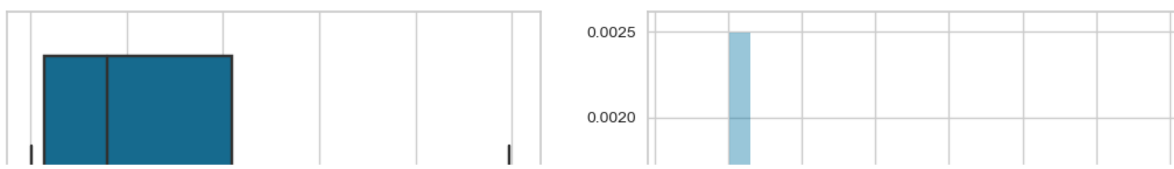
In [50]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2216 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Education            2216 non-null   object 
 1   Income               2216 non-null   float64
 2   Kidhome              2216 non-null   int64  
 3   Teenhome             2216 non-null   int64  
 4   Recency              2216 non-null   int64  
 5   MntWines             2216 non-null   int64  
 6   MntFruits            2216 non-null   int64  
 7   MntMeatProducts      2216 non-null   int64  
 8   MntFishProducts      2216 non-null   int64  
 9   MntSweetProducts     2216 non-null   int64  
 10  MntGoldProds         2216 non-null   int64  
 11  NumDealsPurchases    2216 non-null   int64  
 12  NumWebPurchases      2216 non-null   int64  
 13  NumCatalogPurchases  2216 non-null   int64  
 14  NumStorePurchases    2216 non-null   int64  
 15  NumWebVisitsMonth    2216 non-null   int64  
 16  AcceptedCmp3         2216 non-null   int64  
 17  AcceptedCmp4         2216 non-null   int64  
 18  AcceptedCmp5         2216 non-null   int64  
 19  AcceptedCmp1         2216 non-null   int64  
 20  AcceptedCmp2         2216 non-null   int64  
 21  Complain             2216 non-null   int64  
 22  Response             2216 non-null   int64  
 23  Age                  2216 non-null   int64  
 24  Total Spent          2216 non-null   int64  
 25  Living_With          2216 non-null   object 
 26  Children             2216 non-null   int64  
 27  Family_Size          2216 non-null   int64  
 28  Is_Parent            2216 non-null   int32  
dtypes: float64(1), int32(1), int64(25), object(2)
memory usage: 575.3+ KB
```

In [51]:
```
ot=df[['Total Spent','Income','Age']] # Finding Outliers of These Three as these are the main feature
```

In [52]:
```
# Lets check outliers

for i in ot:
    print(i)
    print("Skewness : ", round(df[i].skew(),3))
    plt.figure(figsize=(13,5))
    plt.subplot(1,2,1)
    sns.boxplot(df[i])
    plt.ylabel('count')
    plt.subplot(1,2,2)
    sns.distplot(x=df[i])
    plt.show()
```

```
Total Spent
Skewness :  0.858

D:\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following varia
ble as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, an
d passing other arguments without an explicit keyword will result in an error or misinterpretatio
n.
  warnings.warn(
D:\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a depr
ecated function and will be removed in a future version. Please adapt your code to use either `di
splot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function f
or histograms).
  warnings.warn(msg, FutureWarning)
```



Treatment of Outliers

In [53]:
```
def detect_outliers(s):
      for i in s:
        Q3, Q1 = np.percentile(df[i], [75 ,25])
        IQR = Q3 - Q1

        ul = Q3+1.5*IQR
        ll = Q1-1.5*IQR

        outliers = df[i][(df[i] > ul) | (df[i] < ll)]
        print(f'*** {i} outlier points***', '\n', outliers, '\n')
```

In [54]:
```
detect_outliers(ot)
```

```
*** Total Spent outlier points***
 1179    2525
1492    2524
1572    2525
Name: Total Spent, dtype: int64

*** Income outlier points***
 164     157243.0
617     162397.0
655     153924.0
687     160803.0
1300    157733.0
1653    157146.0
2132    156924.0
2233    666666.0
Name: Income, dtype: float64

*** Age outlier points***
 192    115
239    122
339    116
Name: Age, dtype: int64
```

In [55]:
```
df= df[(df['Age']<100)]  # Treatment of Outlier Points in "Age"
```

In [56]:
```
df=df[(df["Income"]<600000)] # As in Income Box Plot Max Thresshold is Shown is 600000
```

In [57]:
```
# As in Total Spent Box Plot Max Thresshold is Shown is 2500 we are not treating considering as it car
```

Checking Data Shape

In [58]:
```
df.shape
```

Out[58]: (2212, 29)

In [59]:
```
df.size
```

Out[59]: 64148

Lets Redefine Categorical Values

In [60]:
```
cat = [var for var in df.columns if df[var].dtype=='O']
```

In [61]:
```
# check the number of different labels
for var in cat:
    print(df[var].value_counts() / np.float(len(df)))
    print()
    print()
```

```
Graduate          0.504069
Postgraduate      0.382007
Undergraduate     0.113924
Name: Education, dtype: float64


Partner    0.64557
Alone      0.35443
Name: Living_With, dtype: float64



C:\Users\Admin\AppData\Local\Temp\ipykernel_15144\2310594379.py:3: DeprecationWarning: `np.float` is
a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing th
is will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `
np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-no
tes.html#deprecations (https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations)
  print(df[var].value_counts() / np.float(len(df)))
C:\Users\Admin\AppData\Local\Temp\ipykernel_15144\2310594379.py:3: DeprecationWarning: `np.float` is
a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing th
is will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `
np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-no
tes.html#deprecations (https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations)
  print(df[var].value_counts() / np.float(len(df)))
```

One Hot Encoding

In [62]:
```
list(np.unique(cat))
```

Out[62]:
```
['Education', 'Living_With']
```

In [63]:
```
df['Living_With'].unique()  # Encoding of Living_With Column
```

Out[63]:
```
array(['Alone', 'Partner'], dtype=object)
```

In [64]:
```
df['Living_With'] = df['Living_With'].map({'Alone':0,'Partner':1}) # Replacing the Value
```

In [65]:
```
df['Education'].unique()  # Encoding of Education Column
```

Out[65]:
```
array(['Graduate', 'Postgraduate', 'Undergraduate'], dtype=object)
```

In [66]:
```
df['Education'] = df['Education'].map({'Undergraduate':0,'Graduate':1, 'Postgraduate':2}) # Replacing
```

Checking Data Type

In [67]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2212 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Education            2212 non-null   int64  
 1   Income               2212 non-null   float64
 2   Kidhome              2212 non-null   int64  
 3   Teenhome             2212 non-null   int64  
 4   Recency              2212 non-null   int64  
 5   MntWines             2212 non-null   int64  
 6   MntFruits            2212 non-null   int64  
 7   MntMeatProducts      2212 non-null   int64  
 8   MntFishProducts      2212 non-null   int64  
 9   MntSweetProducts     2212 non-null   int64  
 10  MntGoldProds         2212 non-null   int64  
 11  NumDealsPurchases    2212 non-null   int64  
 12  NumWebPurchases      2212 non-null   int64  
 13  NumCatalogPurchases  2212 non-null   int64  
 14  NumStorePurchases    2212 non-null   int64  
 15  NumWebVisitsMonth    2212 non-null   int64  
 16  AcceptedCmp3         2212 non-null   int64  
 17  AcceptedCmp4         2212 non-null   int64  
 18  AcceptedCmp5         2212 non-null   int64  
 19  AcceptedCmp1         2212 non-null   int64  
 20  AcceptedCmp2         2212 non-null   int64  
 21  Complain             2212 non-null   int64  
 22  Response             2212 non-null   int64  
 23  Age                  2212 non-null   int64  
 24  Total Spent          2212 non-null   int64  
 25  Living_With          2212 non-null   int64  
 26  Children             2212 non-null   int64  
 27  Family_Size          2212 non-null   int64  
 28  Is_Parent            2212 non-null   int32  
dtypes: float64(1), int32(1), int64(27)
memory usage: 509.8 KB
```

As Living_With and Education has been Converted Lets Check Head

In [68]:
```
df.head()
```

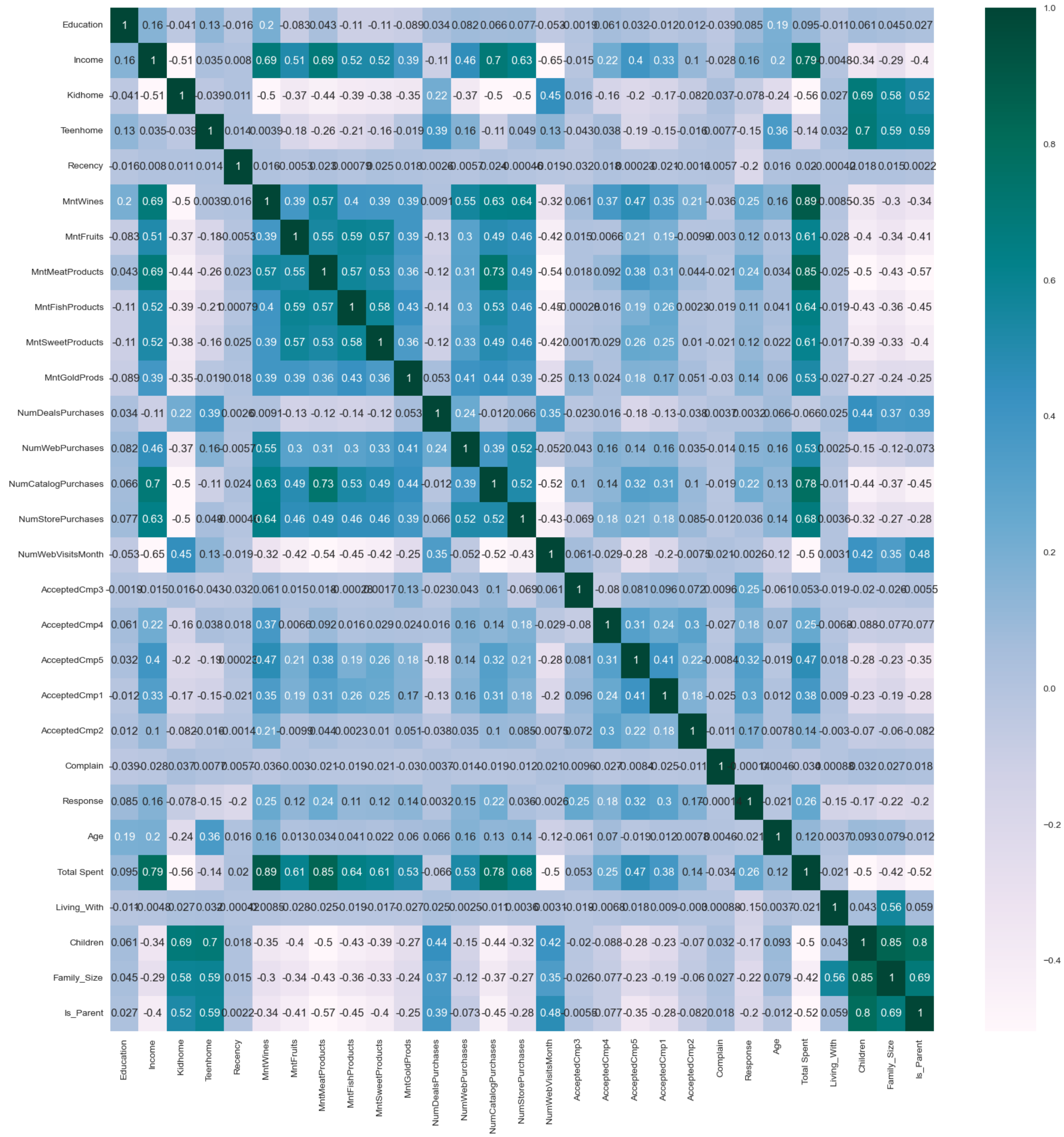
Out[68]:

	Education	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetPro
0	1	58138.0	0	0	58	635	88	546	172	
1	1	46344.0	1	1	38	11	1	6	2	
2	1	71613.0	0	0	26	426	49	127	111	
3	1	26646.0	1	0	26	11	4	20	10	
4	2	58293.0	1	0	94	173	43	118	46	

5 rows × 29 columns

Droping Highly Correlated Features/Columns

```
In [69]: plt.figure(figsize=(20, 20))
         sns.heatmap(data=df.corr(), annot=True,robust=True,cmap="PuBuGn",)
         plt.show()
```



Numeric Features Scaling

```
In [70]: df_old = df.copy()
```

```
In [71]: # creating a subset of dataframe by dropping the features on deals accepted and promotions
         cd = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1','AcceptedCmp2', 'Complain', 'Res
         df = df.drop(cd, axis=1)
```

In [72]:
```python
scaler = StandardScaler()
df = pd.DataFrame(scaler.fit_transform(df), columns = df.columns)
```

Lets Check The Data

In [73]:
```python
df.head()
```

Out[73]:

	Education	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetl
0	-0.411675	0.287105	-0.822754	-0.929699	0.310353	0.977660	1.552041	1.690293	2.453472	
1	-0.411675	-0.260882	1.040021	0.908097	-0.380813	-0.872618	-0.637461	-0.718230	-0.651004	-
2	-0.411675	0.913196	-0.822754	-0.929699	-0.795514	0.357935	0.570540	-0.178542	1.339513	-
3	-0.411675	-1.176114	1.040021	-0.929699	-0.795514	-0.872618	-0.561961	-0.655787	-0.504911	-
4	1.123949	0.294307	1.040021	-0.929699	1.554453	-0.392257	0.419540	-0.218684	0.152508	-

5 rows × 22 columns

As You Can Check Scalling Has Already Been Done

Unsupervised ML - PCA Introduction to Data

In [96]:
```python
pca= PCA(n_components=3,svd_solver='auto',iterated_power='auto',random_state=42) # this will reduce d
pca.fit(df)
```

Out[96]:
```
PCA(n_components=3, random_state=42)
```

In [97]:
```python
s = pca.components_.T
s
```

Out[97]:
```
array([[ 0.01327018,  0.11653055, -0.5174225 ],
       [ 0.27843477,  0.13257917, -0.08785003],
       [-0.24468999,  0.00882847,  0.27571834],
       [-0.08210014,  0.4386403 , -0.15976839],
       [ 0.00331542,  0.01063313,  0.03169025],
       [ 0.2570486 ,  0.17148792, -0.11728232],
       [ 0.23225259, -0.02070385,  0.24868016],
       [ 0.27668547, -0.03478109,  0.06637923],
       [ 0.24170001, -0.03333912,  0.24774682],
       [ 0.23132579, -0.01144468,  0.2521748 ],
       [ 0.18920065,  0.10290936,  0.20694171],
       [-0.06202137,  0.35167307,  0.16439327],
       [ 0.17782304,  0.28475456,  0.03685689],
       [ 0.27380506,  0.06575605,  0.01011379],
       [ 0.245355  ,  0.18039313, -0.00593915],
       [-0.21761359,  0.07920518,  0.11046747],
       [ 0.04199876,  0.19772715, -0.44200857],
       [ 0.31619848,  0.08671832,  0.03292777],
       [-0.02294729,  0.11701385,  0.27733758],
       [-0.23492832,  0.32486048,  0.08151657],
       [-0.20640296,  0.33044023,  0.21383564],
       [-0.22451824,  0.31372319,  0.0841781 ],
       [-0.20714808, -0.33468496, -0.02557589]])
```

In [98]:
```
pd.DataFrame(s, index=df.columns, columns=['Col 1','Col 2', 'col3'])
```

Out[98]:

	Col 1	Col 2	col3
Education	0.013270	0.116531	-0.517423
Income	0.278435	0.132579	-0.087850
Kidhome	-0.244690	0.008828	0.275718
Teenhome	-0.082100	0.438640	-0.159768
Recency	0.003315	0.010633	0.031690
MntWines	0.257049	0.171488	-0.117282
MntFruits	0.232253	-0.020704	0.248680
MntMeatProducts	0.276685	-0.034781	0.066379
MntFishProducts	0.241700	-0.033339	0.247747
MntSweetProducts	0.231326	-0.011445	0.252175
MntGoldProds	0.189201	0.102909	0.206942
NumDealsPurchases	-0.062021	0.351673	0.164393
NumWebPurchases	0.177823	0.284755	0.036857
NumCatalogPurchases	0.273805	0.065756	0.010114
NumStorePurchases	0.245355	0.180393	-0.005939
NumWebVisitsMonth	-0.217614	0.079205	0.110467
Age	0.041999	0.197727	-0.442009
Total Spent	0.316198	0.086718	0.032928
Living_With	-0.022947	0.117014	0.277338
Children	-0.234928	0.324860	0.081517
Family_Size	-0.206403	0.330440	0.213836
Is_Parent	-0.224518	0.313723	0.084178
Clusters	-0.207148	-0.334685	-0.025576

In [99]:
```
pca.explained_variance_
```

Out[99]:
```
array([8.60225892, 3.24182266, 1.43098936])
```

In [100]:
```
pca.explained_variance_ratio_
```

Out[100]:
```
array([0.3715647 , 0.14002681, 0.06180994])
```

In [101]:
```
pd.DataFrame(pca.explained_variance_ratio_, index=range(1,4), columns=['Explained Variability'])
```

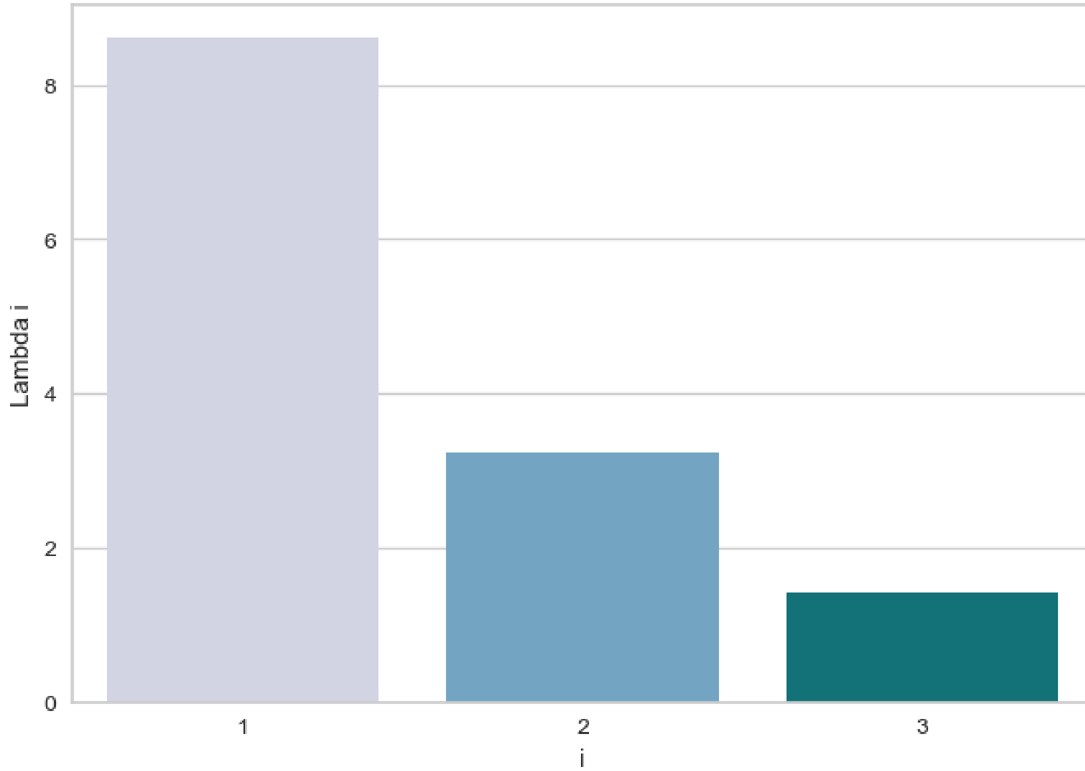
Out[101]:

	Explained Variability
1	0.371565
2	0.140027
3	0.061810

In [102]:
```
pca.explained_variance_ratio_.cumsum()
```

Out[102]:
```
array([0.3715647 , 0.5115915 , 0.57340144])
```

In [103]:
```
sns.barplot(x = list(range(1,4)), y = pca.explained_variance_, palette = 'PuBuGn')
plt.xlabel('i')
plt.ylabel('Lambda i');
```



In [104]:
```
df_PCA = pd.DataFrame(pca.transform(df), columns=(['col1', 'col2', 'col3']))
```

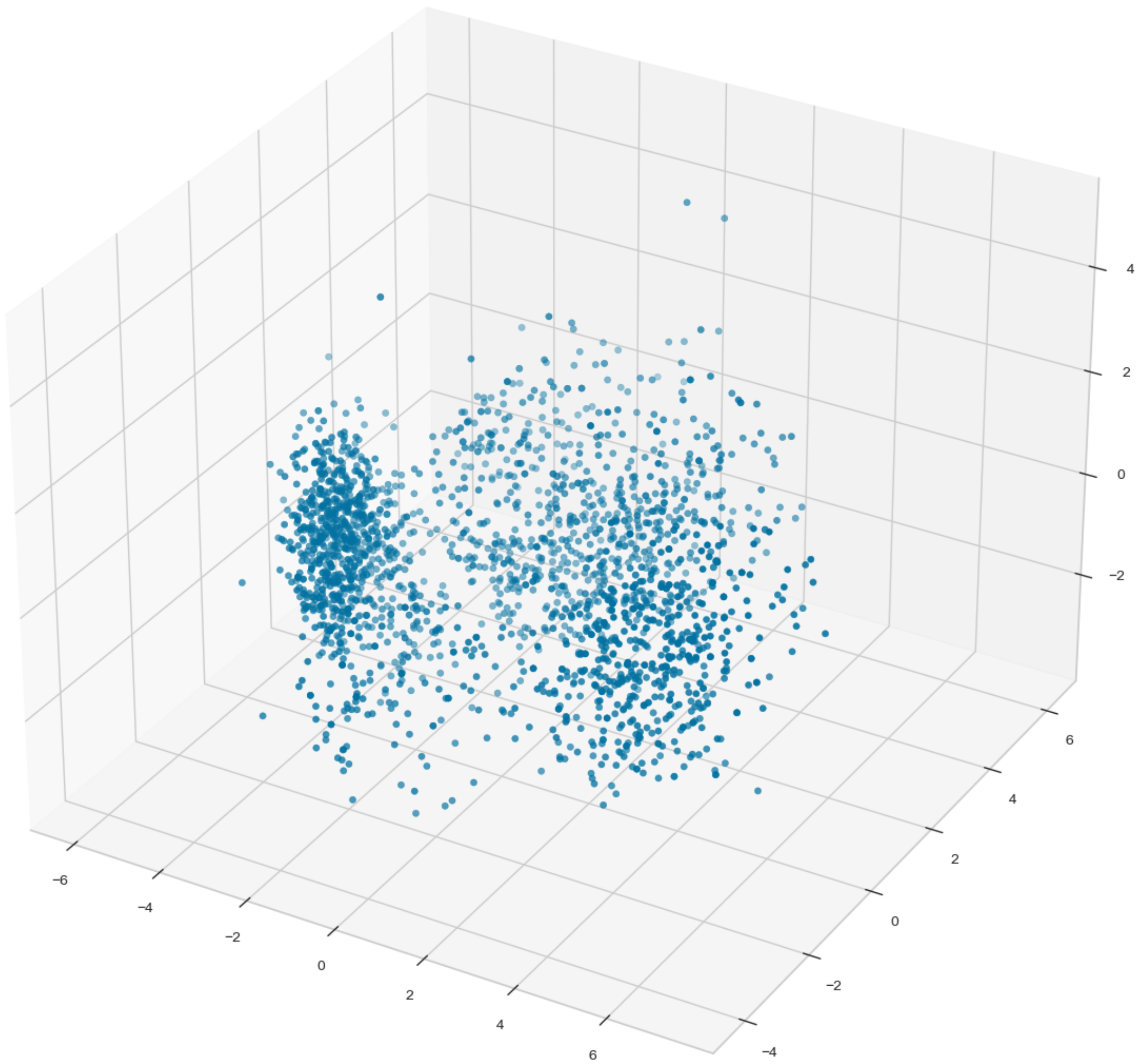
In [105]:
```
df_PCA.describe().T
```

Out[105]:

	count	mean	std	min	25%	50%	75%	max
col1	2212.0	1.847026e-17	2.932961	-6.050912	-2.652635	-0.598926	2.536080	7.317690
col2	2212.0	-4.316419e-18	1.800506	-4.203014	-1.384525	-0.333640	1.418135	6.288027
col3	2212.0	-4.712927e-17	1.196240	-3.517778	-0.857769	-0.020621	0.827002	5.065657

In [106]:
```
x = df_PCA['col1']
y = df_PCA['col2']
z = df_PCA['col3']

fig = plt.figure(figsize=(15,15))
a = fig.add_subplot(111, projection='3d')
a.scatter(x,y,z,  marker='o',depthshade=True)
a.set_title('3D Projection of Data of PCA')
plt.show()
```

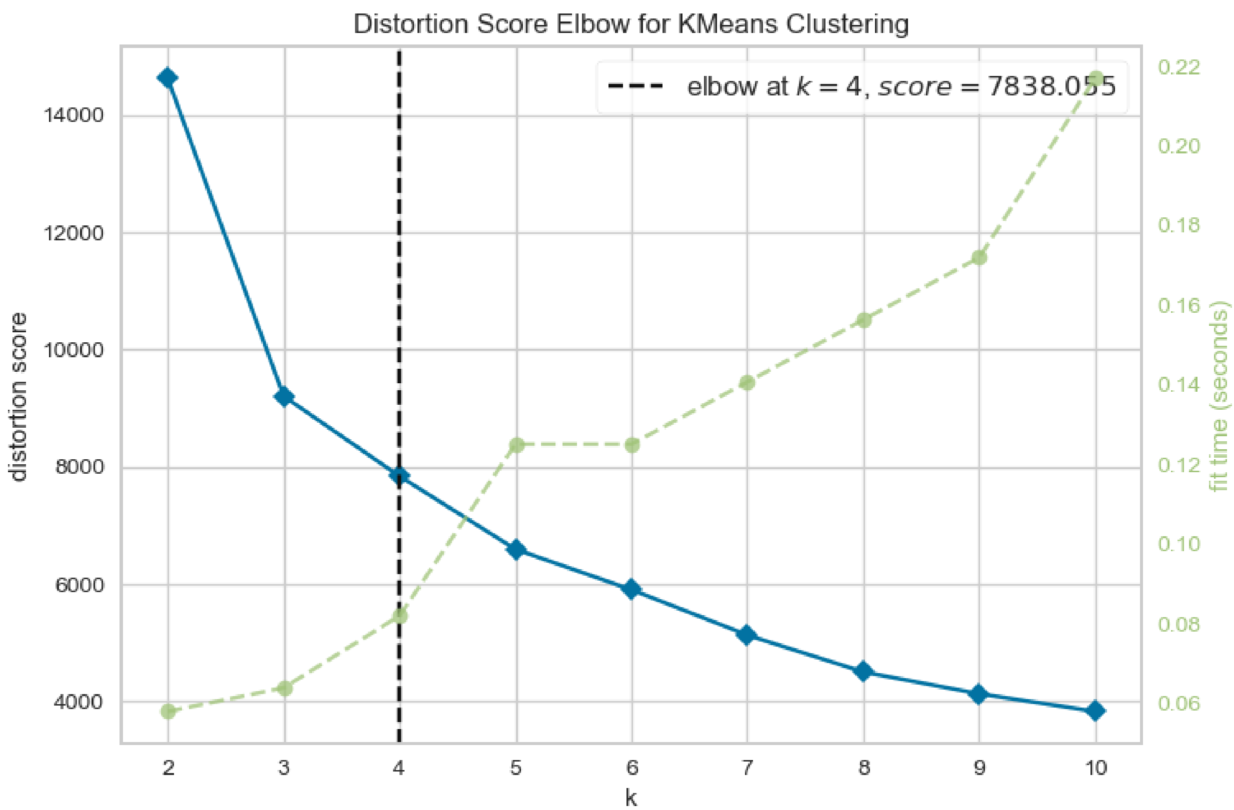
3D Projection of Data of PCA



Clustering

Using Elbow Methord To Determine Number of Cluster Needed for this Data

In [107]:
```
Elbow = KElbowVisualizer(KMeans(n_clusters=4,init='k-means++',n_init=10,max_iter=300,tol=0.0001,randoı
Elbow.fit(df_PCA)
Elbow.show()
```



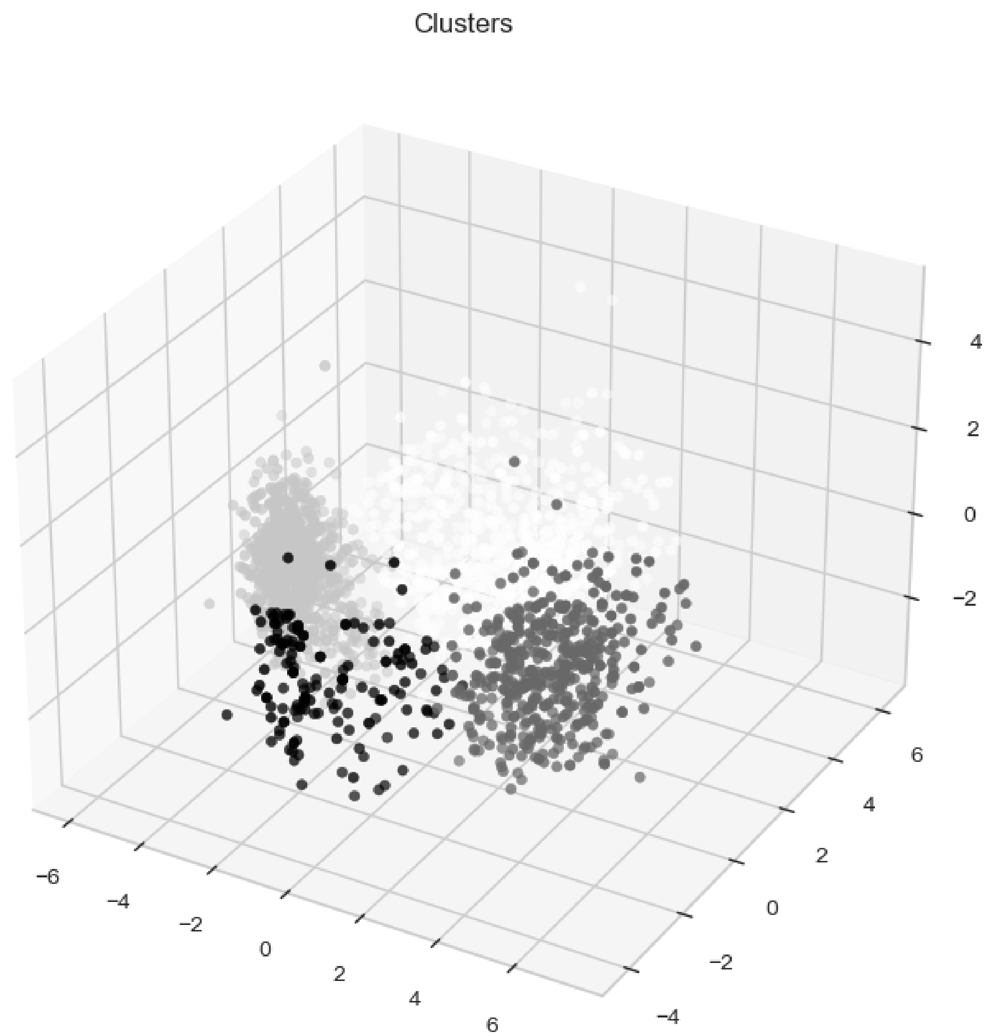
Out[107]: <AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>

In [108]:
```
AC = AgglomerativeClustering(n_clusters=4,affinity='euclidean',compute_full_tree='auto',linkage='ward
# fit model and predict clusters
y_AC = AC.fit_predict(df_PCA)
df_PCA['Clusters'] = y_AC
#Adding the Clusters feature to the orignal dataframe.
df['Clusters'] = y_AC
df_old['Clusters'] = y_AC
```

In [109]:
```
from sklearn.metrics import silhouette_score
# Calculate silhouette score for clusters
score = silhouette_score(df_PCA, y_AC)

score
```

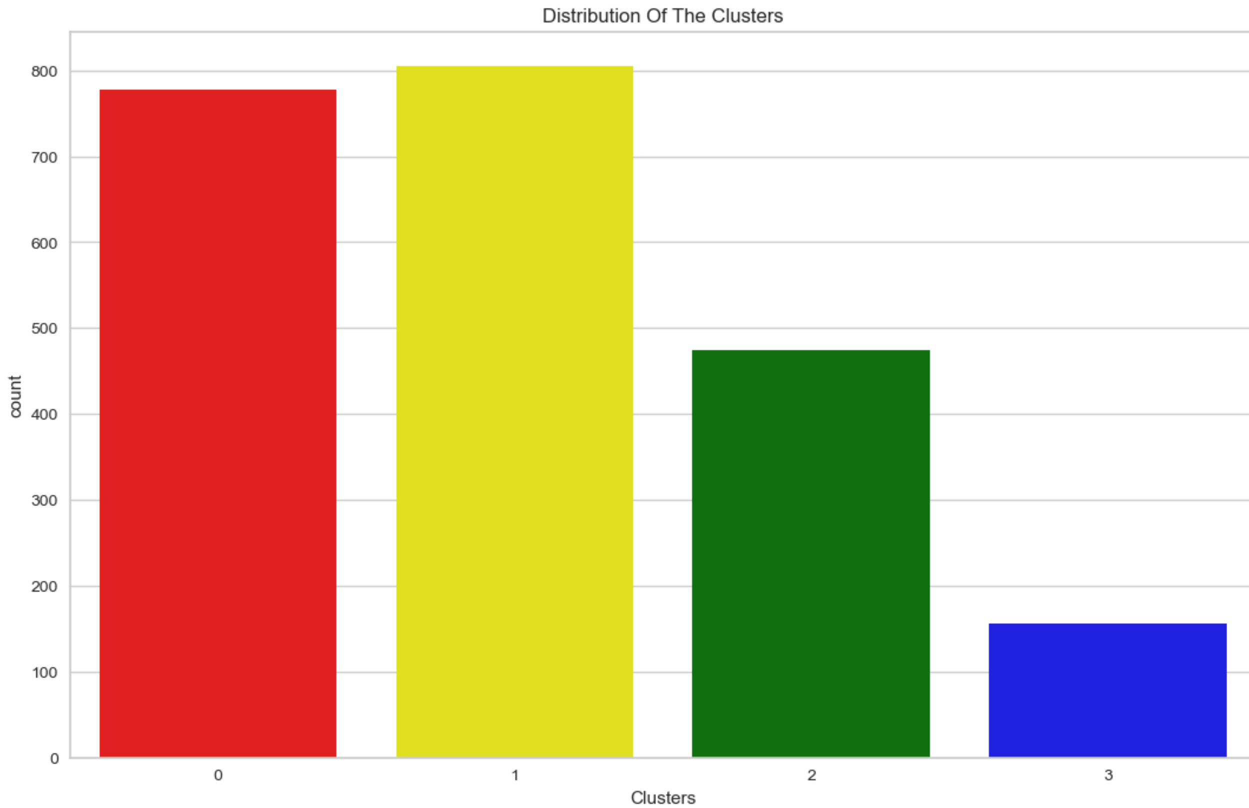
Out[109]: 0.4541150803610546

In [116]:
```
fig = plt.figure(figsize=(13,8))
ax = plt.subplot(111, projection='3d', label='bla')
ax.scatter(x, y, z, c=df_PCA['Clusters'])
ax.set_title('Clusters')
plt.show()
```

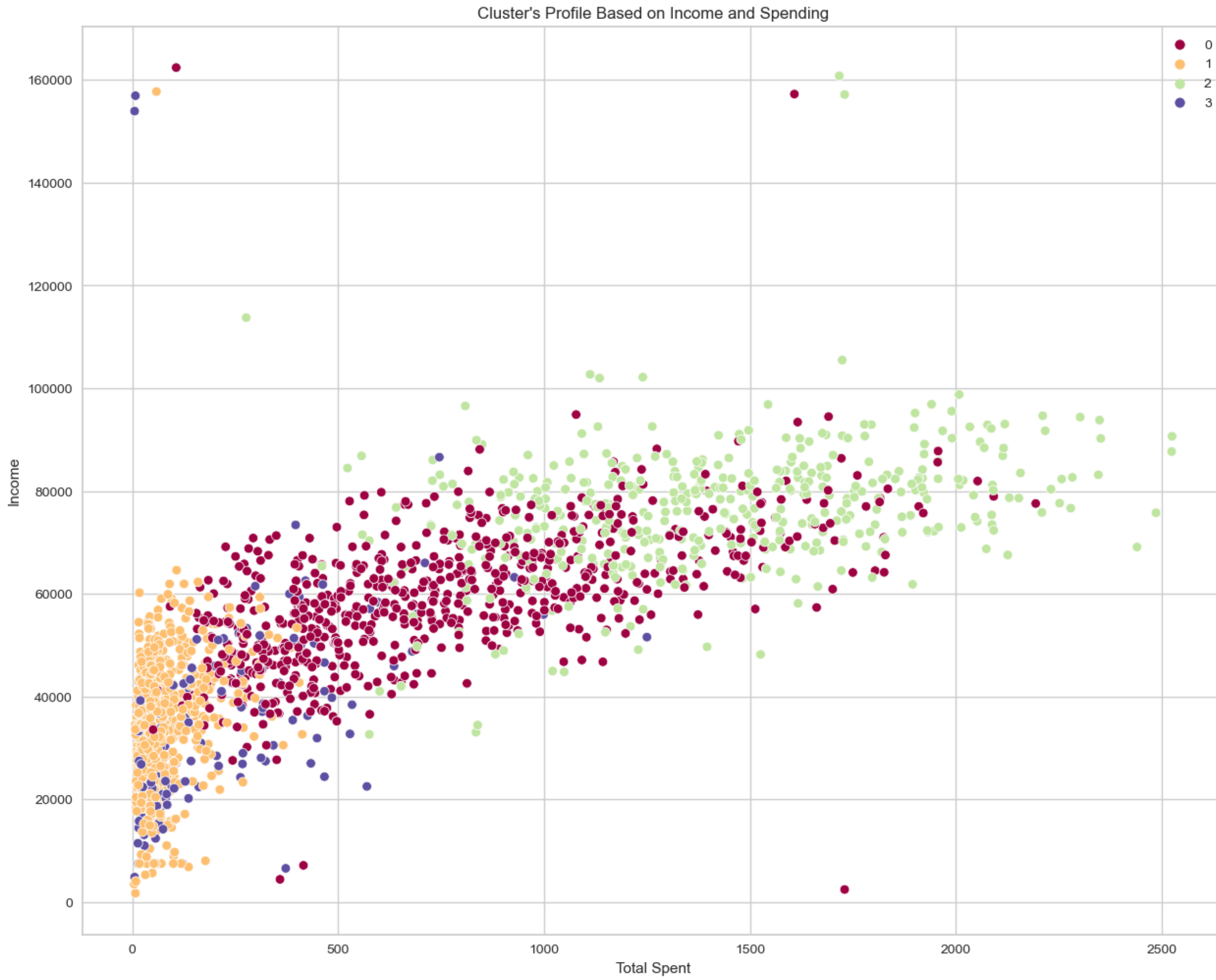


Checking Clusters Formed

In [117]:
```
colour = ['red','yellow', 'green','blue']
plt.figure(figsize=(13,8))
ccf=sns.countplot(x=df['Clusters'], palette= colour)
ccf.set_title('Distribution Of The Clusters')
plt.show()
```



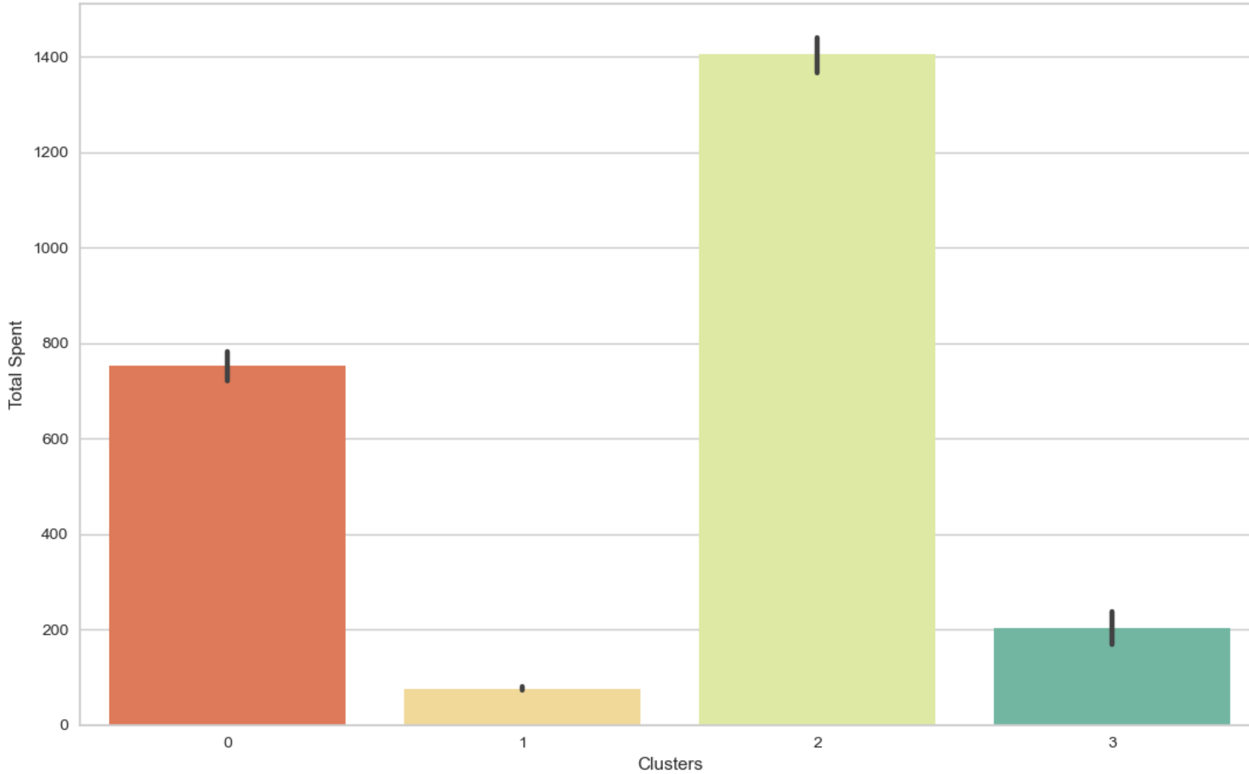
In [118]:
```
plt.figure(figsize=(15,12))
pl = sns.scatterplot(data=df_old, x=df_old['Total Spent'], y=df_old['Income'], hue=df_old['Clusters']
pl.set_title("Cluster's Profile Based on Income and Spending")
plt.legend();
```



Income vs spending plot shows the clusters pattern

In [119]:

```
plt.figure(figsize=(13,8))
sns.barplot(x=df_old['Clusters'], y=df_old['Total Spent'], palette="Spectral")
plt.show();
```



As You Can See That

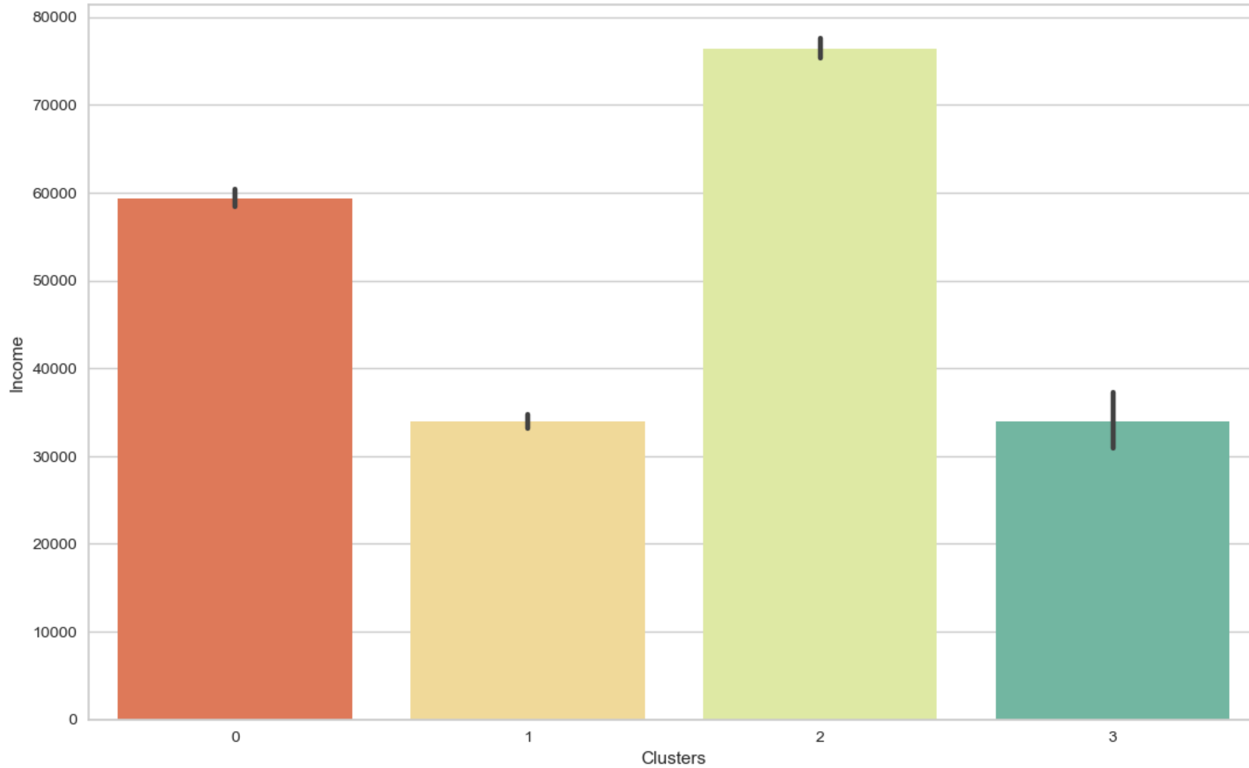
Cluster 1 Has Higher Spending

Cluster 0 Has Average Spending

Cluster 2 Has Low Spending

Cluster 3 Has Lowest Spending

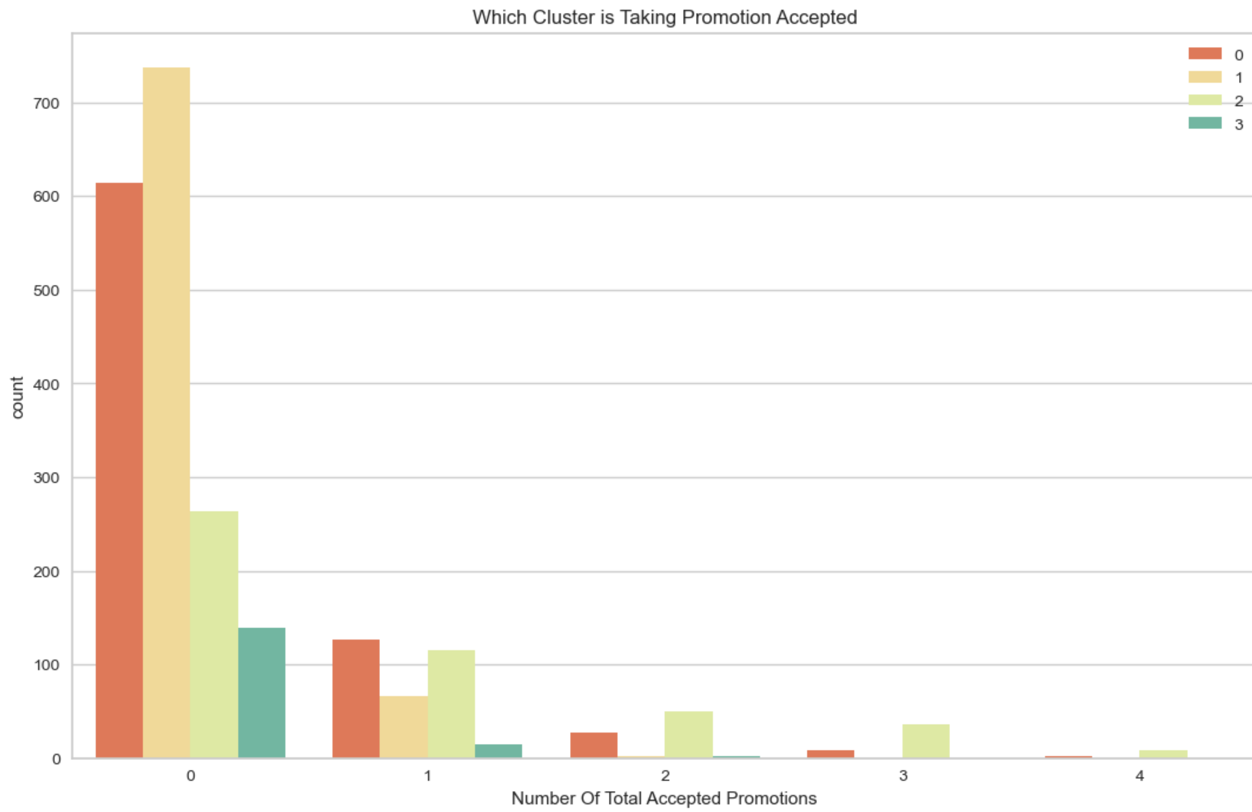
In [120]:
```
plt.figure(figsize=(13,8))
sns.barplot(x=df_old['Clusters'], y=df_old['Income'], palette="Spectral")
plt.show();
```



As You Can See That :

```
Cluster 1 Has Highest Income

Cluster 0 Has High Income

Cluster 2 Has Lowest Among All Income

Cluster 3 Has Low Income But Higher Than Cluster 2
```

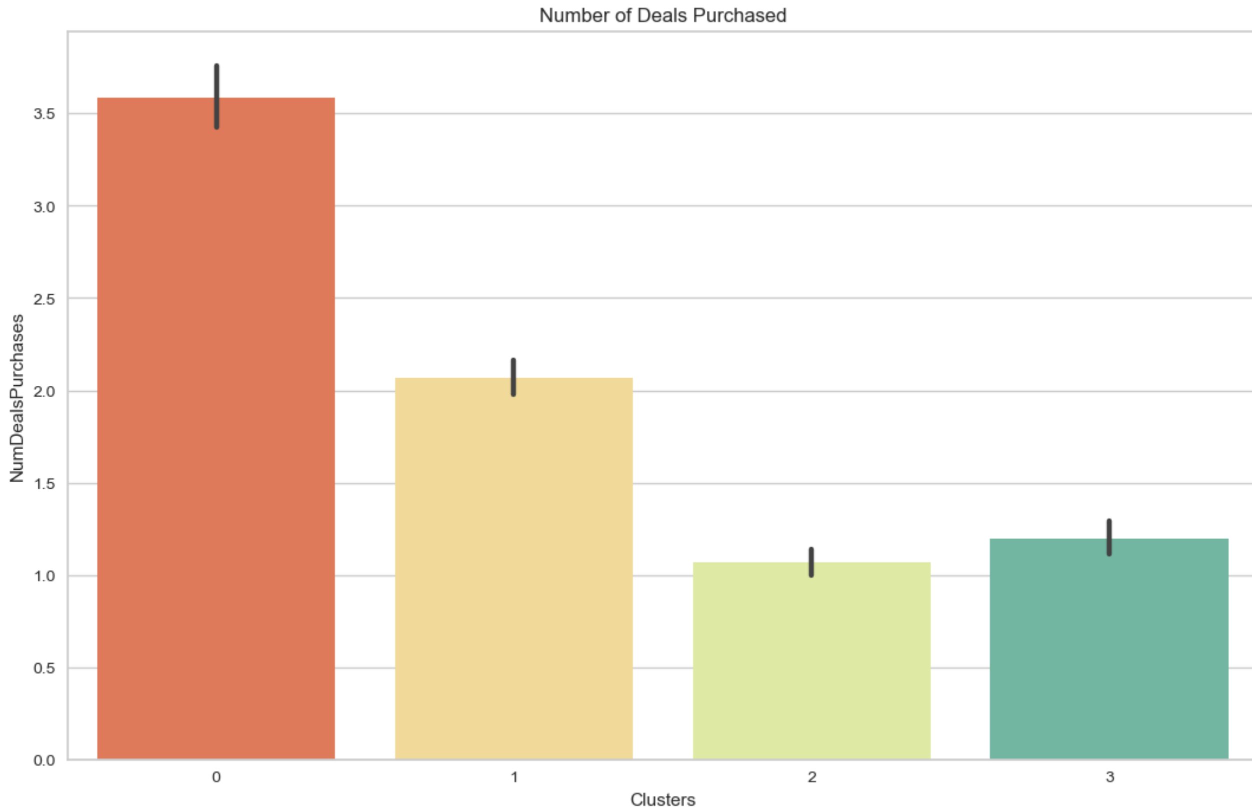
In [121]:
```
df_old['Total_Promos'] = df_old['AcceptedCmp1']+ df_old['AcceptedCmp2']+ df_old['AcceptedCmp3']+ df_o

plt.figure(figsize=(13,8))
cp = sns.countplot(x=df_old['Total_Promos'], hue=df_old['Clusters'], palette= "Spectral")
cp.set_title('Which Cluster is Taking Promotion Accepted')
cp.set_xlabel('Number Of Total Accepted Promotions')
plt.legend(loc='upper right')
plt.show();
```



As You Can See That: Cluster 0 Is Highest Promotion Accepter Whereas Cluster 4 Doesnt Even Care

Hence, There is No One Who is Taking Par in All 5 Promotion Acceptance.

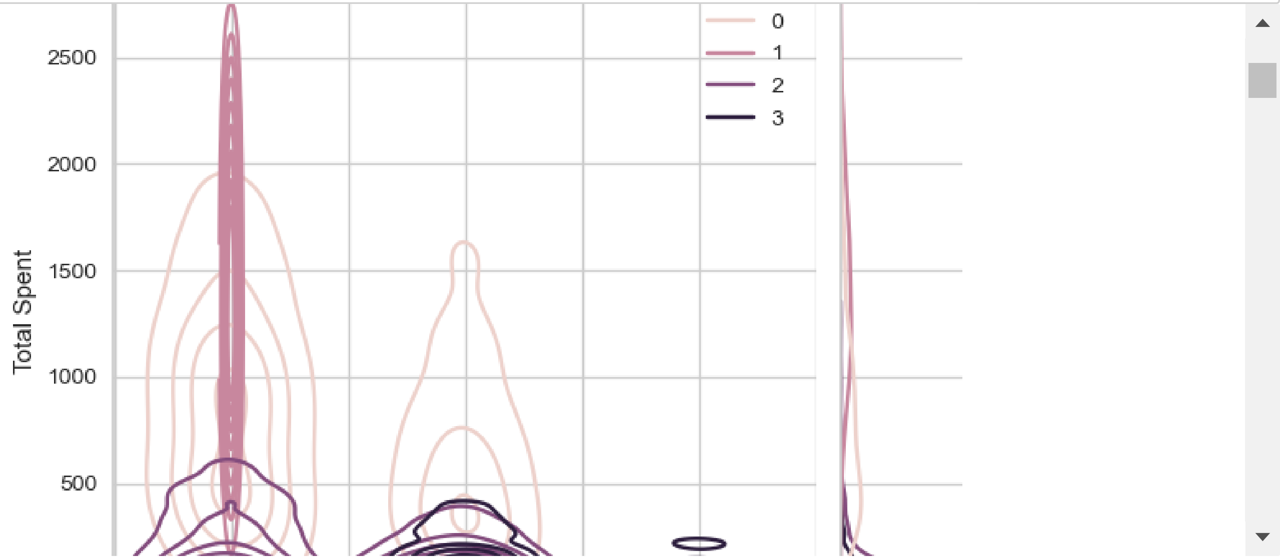
In [122]:
```
plt.figure(figsize=(13,8))
sns.barplot(y=df_old['NumDealsPurchases'],x=df_old['Clusters'], palette= "Spectral")
plt.title('Number of Deals Purchased');
```



As You Can See That:

```
Cluster 0 Purchased Most Number of Deals

Followed By Cluster 3

Followed By Cluster 2

Followed BY Cluster 1
```

In [95]:
```
Personal = ['Kidhome', 'Teenhome', 'Age', 'Children', 'Family_Size', 'Is_Parent', 'Education', 'Livin

for i in Personal:
    plt.figure(figsize=(13,8))
    sns.jointplot(x=df_old[i], y=df_old['Total Spent'], hue=df_old['Clusters'], kind='kde');
```



About Cluster 1:

1. Definitely not having Childred a parent
2. At max are only 2 members in the family.
3. A slight majority of couples over single people
4. Majority are Highly Educated
5. Span all ages from 20 to below 80
6. high income and high spending

About Cluster 3:

1. Definitely a parent
2. At max have 5 members in the family and at least 2
3. Majority of them have a teenager at home
4. Relatively older

About Cluster 2:

1. The majority of these people are parents
2. At max have 3 members in the family
3. They majorly have one kid and typically not tennagers
4. Relatively younger

Cluster 0

1. Definitely a parent
2. At max have 4 members in the family and at least 2
3. Most have a teeanger in home
4. Single parents are a subset of this group
5. Relatively older

```
In [ ]:
```

```
In [ ]:
```