In [ ]:
```
# Import libraries
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from kmodes.kprototypes import KPrototypes
from sklearn.cluster import KMeans
```

In [2]:
```
# import the dataset and Removing the customer id column from the dataset as it provides no useful data, no split for k-mean
```

In [3]:
```
dataset = pd.read_csv("marketing_campaign.csv",sep='\t')
dataset.head()
```

Out[3]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMonth	AcceptedCmp3	Acce
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	58	635	...	7	0	
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	38	11	...	5	0	
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	26	426	...	4	0	
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	26	11	...	6	0	
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	94	173	...	5	0	

5 rows × 29 columns

In [4]:
```
n_row, n_col = dataset.shape
print("The number of rows in dataset are {0} and the number of columns are {1}".format(n_row,n_col))
```

```
The number of rows in dataset are 2240 and the number of columns are 29
```

In [5]:
```
# Information on dataset before removing unnecessary columns
```

In [6]:
```
dataset.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 [7]:
```
dataset.describe(include='all')
```

Out[7]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWe
count	2240.000000	2240.000000	2240	2240	2216.000000	2240.000000	2240.000000	2240	2240.000000	2240.000000	...	
unique	NaN	NaN	5	8	NaN	NaN	NaN	663	NaN	NaN	...	
top	NaN	NaN	Graduation	Married	NaN	NaN	NaN	31-08-2012	NaN	NaN	...	
freq	NaN	NaN	1127	864	NaN	NaN	NaN	12	NaN	NaN	...	
mean	5592.159821	1968.805804	NaN	NaN	52247.251354	0.444196	0.506250	NaN	49.109375	303.935714	...	
std	3246.662198	11.984069	NaN	NaN	25173.076661	0.538398	0.544538	NaN	28.962453	336.597393	...	
min	0.000000	1893.000000	NaN	NaN	1730.000000	0.000000	0.000000	NaN	0.000000	0.000000	...	
25%	2828.250000	1959.000000	NaN	NaN	35303.000000	0.000000	0.000000	NaN	24.000000	23.750000	...	
50%	5458.500000	1970.000000	NaN	NaN	51381.500000	0.000000	0.000000	NaN	49.000000	173.500000	...	
75%	8427.750000	1977.000000	NaN	NaN	68522.000000	1.000000	1.000000	NaN	74.000000	504.250000	...	
max	11191.000000	1996.000000	NaN	NaN	666666.000000	2.000000	2.000000	NaN	99.000000	1493.000000	...	

11 rows × 29 columns

In [8]:
```
# The columns for Z_CostContact and Z_Revenue provide no useful information for my analysis, therefore I remove them.
```

In [9]:
```
dataset.drop(columns = ["ID","Z_CostContact","Z_Revenue"], inplace = True)
```

In [10]:
```
# Information on dataset after removing unnecessary columns
```

In [11]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Year_Birth           2240 non-null   int64
 1   Education            2240 non-null   object
 2   Marital_Status       2240 non-null   object
 3   Income               2216 non-null   float64
 4   Kidhome              2240 non-null   int64
 5   Teenhome             2240 non-null   int64
 6   Dt_Customer          2240 non-null   object
 7   Recency              2240 non-null   int64
 8   MntWines             2240 non-null   int64
 9   MntFruits            2240 non-null   int64
 10  MntMeatProducts      2240 non-null   int64
 11  MntFishProducts      2240 non-null   int64
 12  MntSweetProducts     2240 non-null   int64
 13  MntGoldProds         2240 non-null   int64
 14  NumDealsPurchases    2240 non-null   int64
 15  NumWebPurchases      2240 non-null   int64
 16  NumCatalogPurchases  2240 non-null   int64
 17  NumStorePurchases    2240 non-null   int64
 18  NumWebVisitsMonth    2240 non-null   int64
 19  AcceptedCmp3         2240 non-null   int64
 20  AcceptedCmp4         2240 non-null   int64
 21  AcceptedCmp5         2240 non-null   int64
 22  AcceptedCmp1         2240 non-null   int64
 23  AcceptedCmp2         2240 non-null   int64
 24  Complain             2240 non-null   int64
 25  Response             2240 non-null   int64
dtypes: float64(1), int64(22), object(3)
memory usage: 455.1+ KB
```

In [12]:
```
# we have  a mix data type, Dt_Customer is data type object
# Education and Marital_Status are categorical features
# Income has missing values
```

In [13]:
```
# exploring the categorial features
print(dataset["Marital_Status"].value_counts())
```

```
Married     864
Together    580
Single      480
Divorced    232
Widow        77
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64
```

In [14]:
```
print(dataset["Education"].value_counts())
```

```
Graduation    1127
PhD            486
Master         370
2n Cycle       203
Basic           54
Name: Education, dtype: int64
```

In [15]:
```
# Feature Engineering
```

In [16]:
```
dataset["Marital_Status"].replace({'Alone': 'Single', 'Absurd': 'Single','YOLO':'Single'},inplace=True)
```

In [17]:
```
# replace date of birth with age
```

In [18]:
```
dataset['Year_Birth']= 2022-dataset['Year_Birth']
dataset.rename(columns={"Year_Birth": "Age"},inplace=True)
```

In [19]:
```
dataset.head()
```

Out[19]:

	Age	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	MntFruits	...	NumCatalogPurchases	NumStorePurchases
0	65	Graduation	Single	58138.0	0	0	04-09-2012	58	635	88	...	10	4
1	68	Graduation	Single	46344.0	1	1	08-03-2014	38	11	1	...	1	2
2	57	Graduation	Together	71613.0	0	0	21-08-2013	26	426	49	...	2	10
3	38	Graduation	Together	26646.0	1	0	10-02-2014	26	11	4	...	0	4
4	41	PhD	Married	58293.0	1	0	19-01-2014	94	173	43	...	3	6

5 rows × 26 columns

In [20]:
```
#Total spendings on all items
dataset["Total_Spent"] = dataset["MntWines"]+ dataset["MntFruits"]+ dataset["MntMeatProducts"]+ dataset["MntFishProducts"]+
```

In [21]:
```
# creating a new feature showing the number of days of customer engagement
```

In [22]:
```
dataset['Dt_Customer'] = pd.to_datetime(dataset.Dt_Customer)
newest_customer = dataset['Dt_Customer'].max()
dataset['newest_customer'] = newest_customer
dataset['days_engaged'] = (dataset['newest_customer'] - dataset['Dt_Customer']).dt.days
print(dataset['days_engaged'])
dataset.drop(columns=['Dt_Customer','newest_customer'],inplace=True)
```

```
0       971
1       125
2       472
3        65
4       321
       ...
2235    541
2236     61
2237    315
2238    316
2239    782
Name: days_engaged, Length: 2240, dtype: int64

C:\Users\yasha\AppData\Local\Temp\ipykernel_31952\4042051918.py:1: UserWarning: Parsing '21-08-2013' in DD/MM/YYYY forma
t. Provide format or specify infer_datetime_format=True for consistent parsing.
  dataset['Dt_Customer'] = pd.to_datetime(dataset.Dt_Customer)
C:\Users\yasha\AppData\Local\Temp\ipykernel_31952\4042051918.py:1: UserWarning: Parsing '19-01-2014' in DD/MM/YYYY forma
t. Provide format or specify infer_datetime_format=True for consistent parsing.
  dataset['Dt_Customer'] = pd.to_datetime(dataset.Dt_Customer)
C:\Users\yasha\AppData\Local\Temp\ipykernel_31952\4042051918.py:1: UserWarning: Parsing '13-11-2012' in DD/MM/YYYY forma
```

In [23]:
```
dataset.describe()
```

Out[23]:

	Age	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts
count	2240.000000	2216.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000
mean	53.194196	52247.251354	0.444196	0.506250	49.109375	303.935714	26.302232	166.950000	37.525446	27.062946
std	11.984069	25173.076661	0.538398	0.544538	28.962453	336.597393	39.773434	225.715373	54.628979	41.280498
min	26.000000	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	45.000000	35303.000000	0.000000	0.000000	24.000000	23.750000	1.000000	16.000000	3.000000	1.000000
50%	52.000000	51381.500000	0.000000	0.000000	49.000000	173.500000	8.000000	67.000000	12.000000	8.000000
75%	63.000000	68522.000000	1.000000	1.000000	74.000000	504.250000	33.000000	232.000000	50.000000	33.000000
max	129.000000	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	263.000000

8 rows × 25 columns

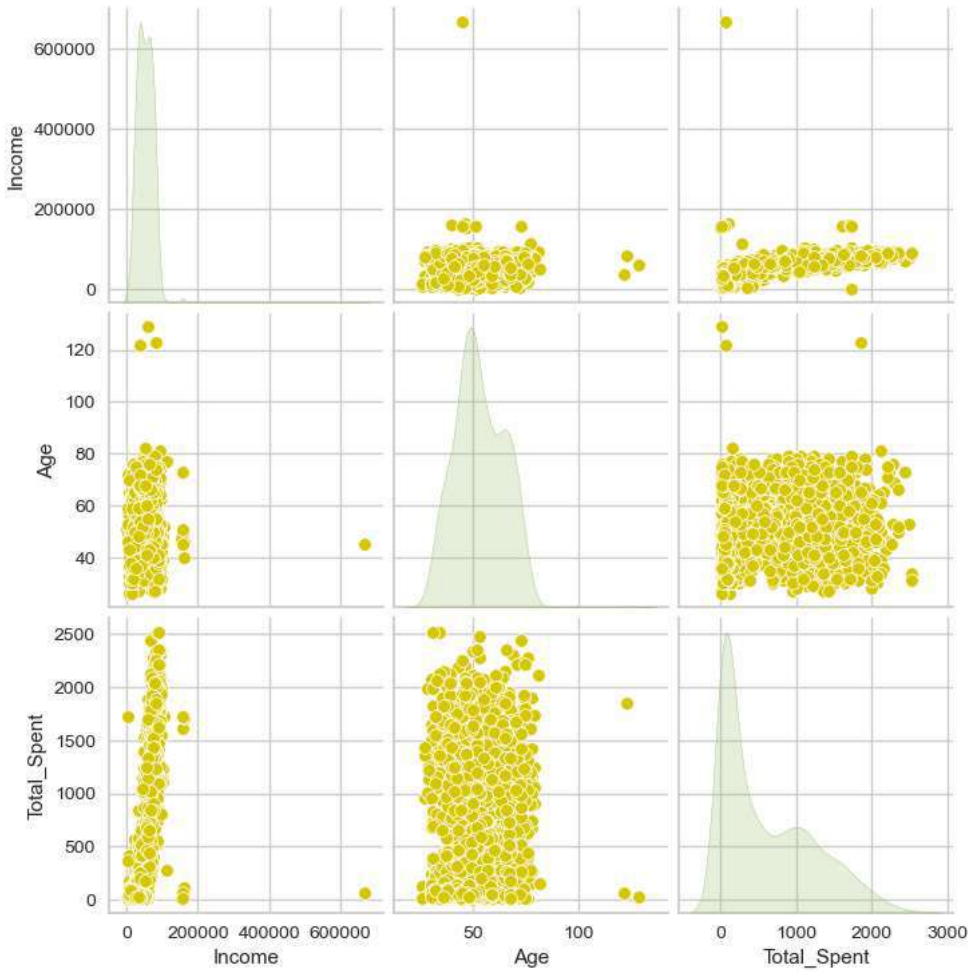
In [24]:
```
# Data Cleaning process in order to target NAN values
```

In [25]:
```
dataset.isna().sum()
dataset.dropna(inplace = True)
```

In [26]:
```
# checking on the relevant features
```

In [27]:
```
plt.figure()
cols_to_plot = ['Income', 'Age', 'Total_Spent']
sns.pairplot(dataset[cols_to_plot], diag_kind='kde', diag_kws={'color':'g'}, plot_kws={'color':'y'})
plt.show()
```

```
<Figure size 800x550 with 0 Axes>
```



In [28]:
```
# Dropping the outliers
```

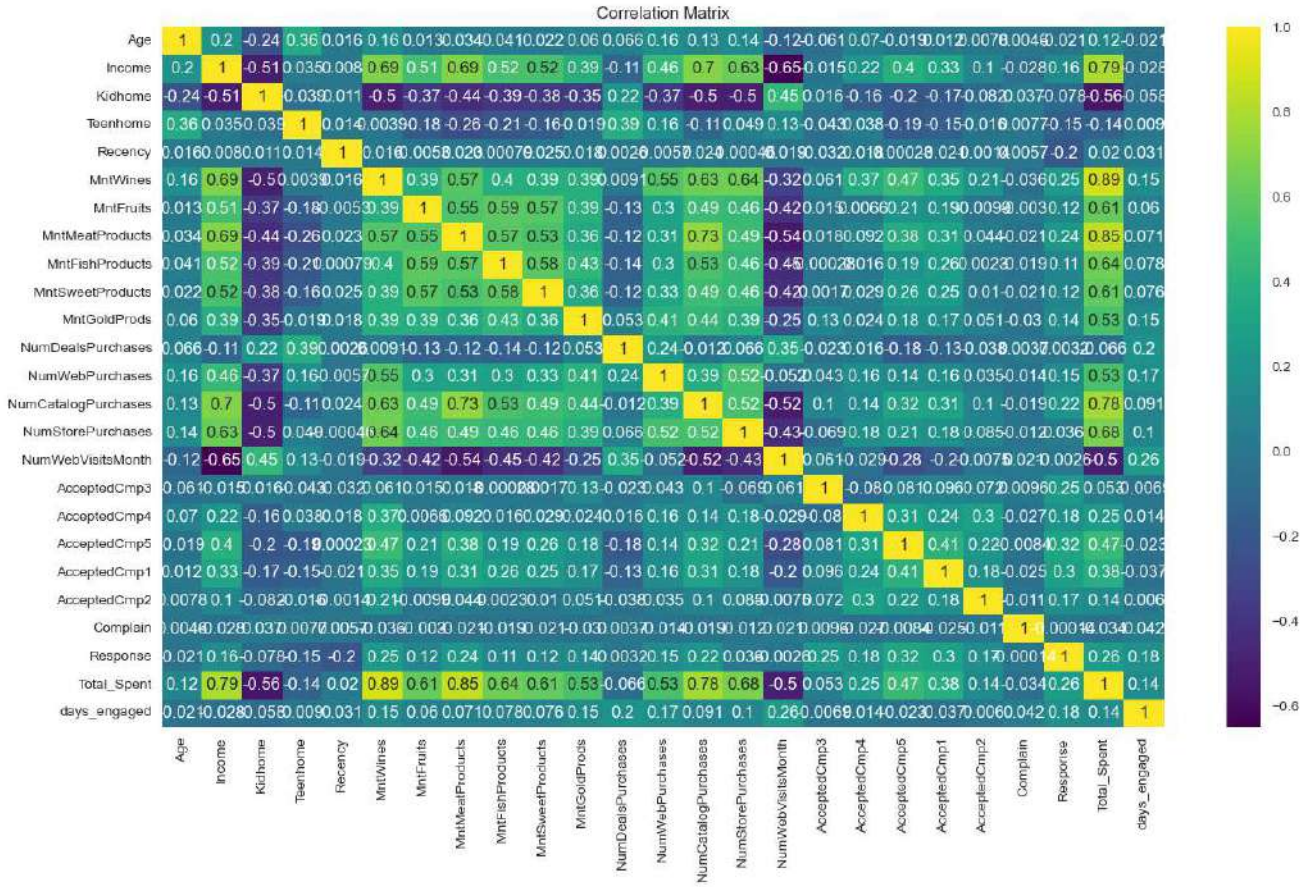
In [29]:
```
dataset = dataset[(dataset["Age"]<100)]
dataset = dataset[(dataset["Income"]<600000)]
```

In [30]:
```
# num of samples after cleaning
print(len(dataset))
```

```
2212
```

In [31]:
```
# Show casing the correlation among the features
```

In [32]:
```
plt.figure(figsize=(16,9))
sns.heatmap(dataset.corr(), cmap='viridis', annot = True)
plt.title('Correlation Matrix')
plt.show()
```



In [33]:
```
# Encoding ordinal features using OrdinalEncoder
```

In [34]:
```
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder, StandardScaler, MinMaxScaler
education_order = ['Basic', '2n Cycle','Graduation','Master', 'PhD']
oe = OrdinalEncoder(categories = [education_order], dtype=int)
education_oe = oe.fit_transform(dataset[['Education']])
dataset_enc= dataset.assign(Education_encode=education_oe)
print(dataset_enc.shape)
print(dataset_enc[['Education', 'Education_encode']])
```

```
(2212, 28)
       Education  Education_encode
0     Graduation                 2
1     Graduation                 2
2     Graduation                 2
3     Graduation                 2
4            PhD                 4
...          ...               ...
2235  Graduation                 2
2236         PhD                 4
2237  Graduation                 2
2238      Master                 3
2239         PhD                 4

[2212 rows x 2 columns]
```

In [35]:
```
ohe =  OneHotEncoder(sparse=False, dtype='int')
Marital_ohe = ohe.fit_transform(dataset[['Marital_Status']])
Marital_ohe = pd.DataFrame(data=Marital_ohe,columns=ohe.get_feature_names(['Marital_Status']), index=dataset.index,)
dataset_enc = pd.concat([dataset_enc,Marital_ohe],axis=1)
dataset_enc.drop(columns=['Marital_Status','Education'],inplace=True)
```

```
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function get_feature_names is d
eprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [36]:
```
dataset_enc.info()
```

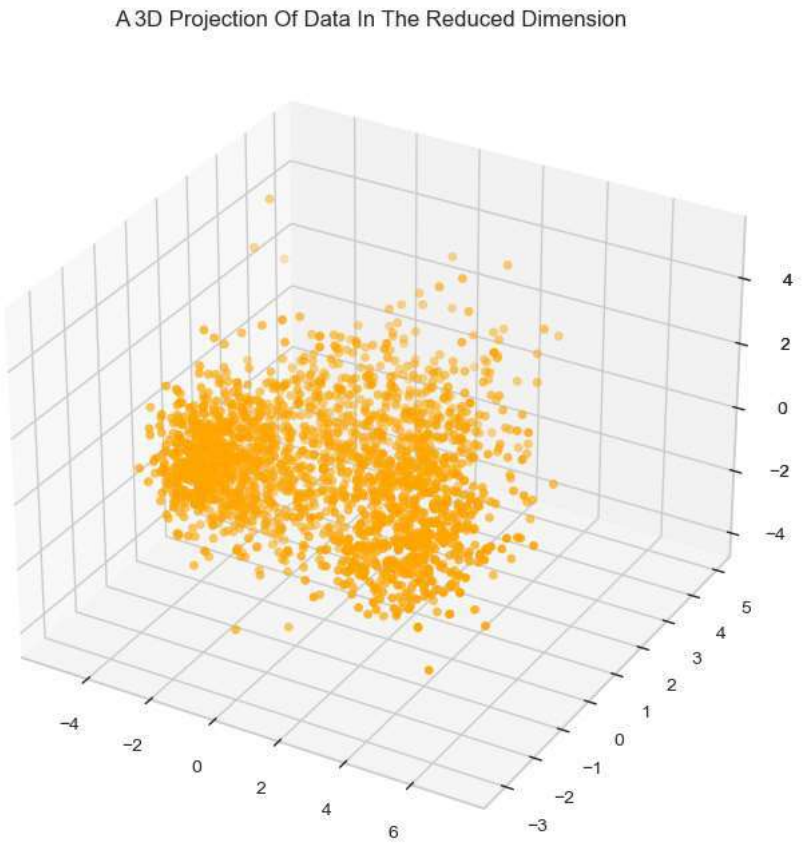
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2212 entries, 0 to 2239
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Age                      2212 non-null   int64
 1   Income                   2212 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  Total_Spent              2212 non-null   int64
 24  days_engaged             2212 non-null   int64
 25  Education_encode         2212 non-null   int32
 26  Marital_Status_Divorced  2212 non-null   int32
 27  Marital_Status_Married   2212 non-null   int32
 28  Marital_Status_Single    2212 non-null   int32
 29  Marital_Status_Together  2212 non-null   int32
 30  Marital_Status_Widow     2212 non-null   int32
dtypes: float64(1), int32(6), int64(24)
memory usage: 501.2 KB
```

In [37]:
```
binary_columns = ['Marital_Status_Divorced','Marital_Status_Married', 'Marital_Status_Single','Marital_Status_Together','Mar
                  ,'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1','AcceptedCmp2', 'Complain', 'Response']
dataset_to_scaler = dataset_enc.drop(columns=binary_columns)
scaler = StandardScaler().fit_transform(dataset_to_scaler)
scaled_dataset = pd.DataFrame(scaler,columns= dataset_to_scaler.columns )
binary_series = dataset_enc[binary_columns]
scaled_dataset = pd.concat([scaled_dataset,binary_series],axis=1)
scaled_dataset.isna().sum()
scaled_dataset = scaled_dataset.fillna(0)
```

In [38]:
```
# Dimensionality reduction with PCA
```

In [39]:
```
from sklearn.compose import ColumnTransformer
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
pca.fit(scaled_dataset)
PCA_dataset = pd.DataFrame(pca.transform(scaled_dataset), columns=(["feature1","feature2", "feature3"]))
```

In [40]:
```
plt.figure(figsize=(10,8))
plt.axes(projection='3d').scatter(PCA_dataset["feature1"], PCA_dataset["feature2"], PCA_dataset["feature3"])
plt.axes(projection='3d').scatter(PCA_dataset["feature1"], PCA_dataset["feature2"], PCA_dataset["feature3"]).set_color('orar
plt.title("A 3D Projection Of Data In The Reduced Dimension")
plt.show()
```

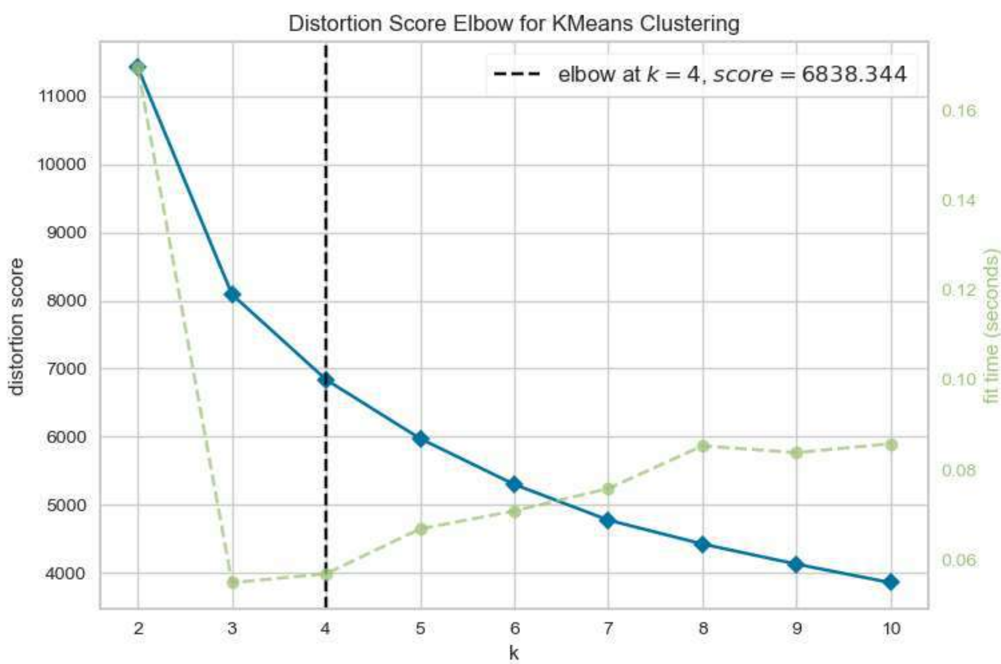


In [41]:
```
# Clustering
```

In [42]:
```
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
```

In [43]:
```
Elbow_M = KElbowVisualizer(KMeans(), k=(2,11))
Elbow_M.fit(PCA_dataset)
Elbow_M.show()
```

```
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=9.
  warnings.warn(
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=9.
  warnings.warn(
```



Out[43]: <AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>

In [44]:
```
# Using elbow curve to find the optimum number of clusters
```

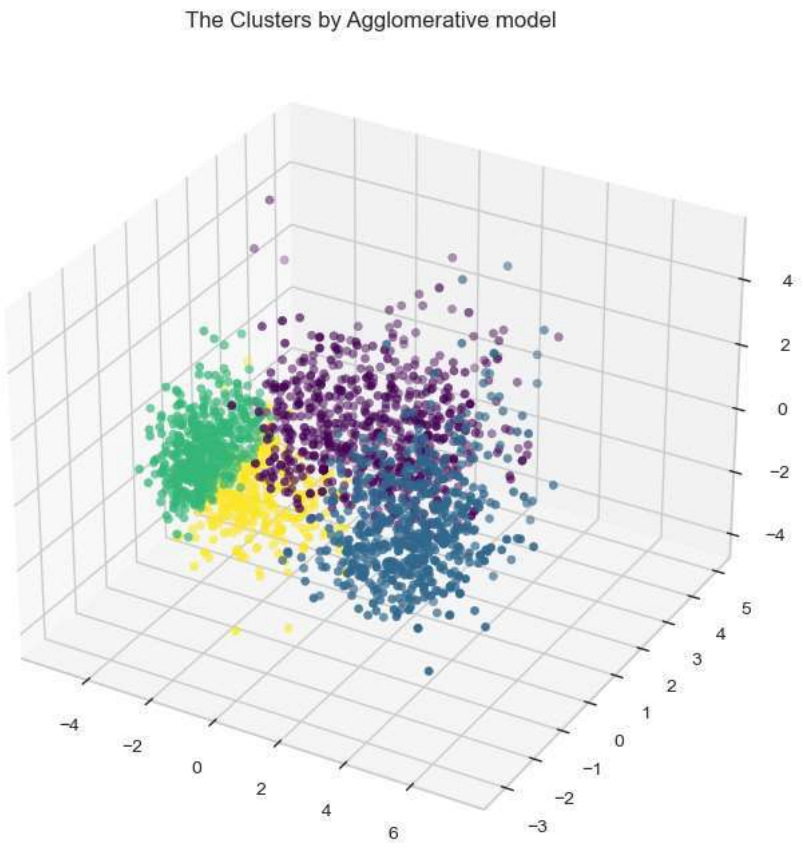
In [45]:
```
# The Agglomerative Clustering model
```

In [46]:
```
AC = AgglomerativeClustering(n_clusters=4)
yhat_AC = AC.fit_predict(PCA_dataset)
PCA_dataset["Clusters"] = yhat_AC
scaled_dataset["Clusters"]= yhat_AC
```

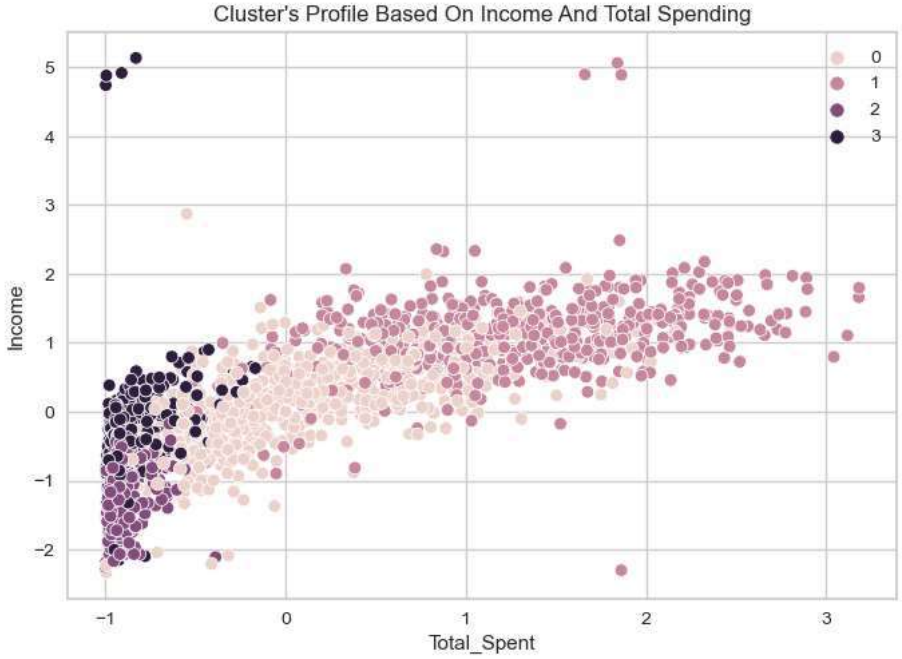
In [47]:
```
# Plotting clusters
```

In [48]:
```
fig = plt.figure(figsize=(10,8))
plt.axes(projection='3d').scatter(PCA_dataset["feature1"], PCA_dataset["feature2"], PCA_dataset["feature3"], c=PCA_dataset['
plt.title("The Clusters by Agglomerative model")
```

Out[48]:
```
Text(0.5, 0.92, 'The Clusters by Agglomerative model')
```



In [49]:
```
catterplot(data = scaled_dataset,x=scaled_dataset["Total_Spent"], y=scaled_dataset["Income"],hue=scaled_dataset["Clusters"])
le("Cluster's Profile Based On Income And Total Spending")
()
```



In [ ]:
```
# clustering using K-means model
```

In [59]:
```
kmeans = KMeans(n_clusters =4 , init = 'k-means++', random_state = 50)
# fit model and predict clusters
labels = kmeans.fit_predict(PCA_dataset)
PCA_dataset["Clusters"] = labels
#Adding the Clusters feature to the orignal dataframe.
scaled_dataset["Clusters"]= labels


#Plotting the clusters
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(PCA_dataset["feature1"], PCA_dataset["feature2"], PCA_dataset["feature3"], s=40, c=PCA_dataset["Clusters"], marke
ax.set_title(" Clustering by K-Means model")
plt.show()
```

```
C:\Users\yasha\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=9.
  warnings.warn(
```