In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
data = pd.read_csv('customer_data.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer
0	5524	1957	Graduation	Single	58138	0	0	04-09-2012
1	2174	1954	Graduation	Single	46344	1	1	08-03-2014
2	4141	1965	Graduation	Together	71613	0	0	21-08-2013
3	6182	1984	Graduation	Together	26646	1	0	10-02-2014
4	5324	1981	PhD	Married	58293	1	0	19-01-2014

5 rows × 29 columns

In [4]:

```
data.tail()
```

Out[4]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Custome
195	9265	1953	Graduation	Married	75027	0	1	09-01-201
196	8867	1988	PhD	Married	67546	0	0	31-08-201
197	8932	1969	Master	Together	65176	0	1	29-10-201
198	10236	1975	Master	Single	31160	1	0	16-09-201
199	6340	1985	Graduation	Single	29938	1	0	27-10-201

5 rows × 29 columns

In [5]:

```
data.shape
```

Out[5]:

```
(200, 29)
```

In [6]:

```
data.columns
```

Out[6]:

```
Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidho
me',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue', 'Respons
e'],
      dtype='object')
```

In [7]:

```
data.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

```
data.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]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   ID                   200 non-null    int64 
 1   Year_Birth           200 non-null    int64 
 2   Education            200 non-null    object
 3   Marital_Status       200 non-null    object
 4   Income               200 non-null    int64 
 5   Kidhome              200 non-null    int64 
 6   Teenhome             200 non-null    int64 
 7   Dt_Customer          200 non-null    object
 8   Recency              200 non-null    int64 
 9   MntWines             200 non-null    int64 
 10  MntFruits            200 non-null    int64 
 11  MntMeatProducts      200 non-null    int64 
 12  MntFishProducts      200 non-null    int64 
 13  MntSweetProducts     200 non-null    int64 
 14  MntGoldProds         200 non-null    int64 
 15  NumDealsPurchases    200 non-null    int64 
 16  NumWebPurchases      200 non-null    int64 
 17  NumCatalogPurchases  200 non-null    int64 
 18  NumStorePurchases    200 non-null    int64 
 19  NumWebVisitsMonth    200 non-null    int64 
 20  AcceptedCmp3         200 non-null    int64 
 21  AcceptedCmp4         200 non-null    int64 
 22  AcceptedCmp5         200 non-null    int64 
 23  AcceptedCmp1         200 non-null    int64 
 24  AcceptedCmp2         200 non-null    int64 
 25  Complain             200 non-null    int64 
 26  Z_CostContact        200 non-null    int64 
 27  Z_Revenue            200 non-null    int64 
 28  Response             200 non-null    int64 
dtypes: int64(26), object(3)
memory usage: 45.4+ KB
```

In [10]:

```
data.describe()
```

Out[10]:

	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	MntW
count	200.000000	200.000	200.000000	200.000000	200.000000	200.000000	200.00(
mean	5633.330000	1968.260	53405.395000	0.460000	0.515000	46.860000	321.00
std	3270.834831	12.064	22701.597661	0.556731	0.566906	27.962832	333.05
min	0.000000	1943.000	2447.000000	0.000000	0.000000	0.000000	0.00(
25%	2681.500000	1958.000	35842.500000	0.000000	0.000000	24.000000	21.50(
50%	6038.000000	1969.000	53857.500000	0.000000	0.000000	47.000000	222.00(
75%	8450.250000	1977.000	69406.000000	1.000000	1.000000	69.000000	524.00(
max	11178.000000	1992.000	157243.000000	2.000000	2.000000	99.000000	1349.00(

8 rows × 26 columns

In [11]:

```
data.nunique()
```

Out[11]:

```
ID                     200
Year_Birth              48
Education                4
Marital_Status           6
Income                 197
Kidhome                  3
Teenhome                 3
Dt_Customer            173
Recency                 85
MntWines               156
MntFruits               71
MntMeatProducts        133
MntFishProducts         77
MntSweetProducts        70
MntGoldProds            85
NumDealsPurchases       12
NumWebPurchases         12
NumCatalogPurchases     14
NumStorePurchases       13
NumWebVisitsMonth       11
AcceptedCmp3             2
AcceptedCmp4             2
AcceptedCmp5             2
AcceptedCmp1             2
AcceptedCmp2             1
Complain                 2
Z_CostContact            1
Z_Revenue                1
Response                 2
dtype: int64
```

In [12]:

```
data["Dt_Customer"] = pd.to_datetime(data["Dt_Customer"])
```

In [13]:

```
dates = []
for i in data["Dt_Customer"]:
    i = i.date()
    dates.append(i)
#Dates of the newest and oldest recorded customer
print("The newest customer's enrolment date in therecords:",max(dates))
print("The oldest customer's enrolment date in the records:",min(dates))
```

```
The newest customer's enrolment date in therecords: 2014-12-05
The oldest customer's enrolment date in the records: 2012-02-11
```

In [14]:

```
days = []
d1 = max(dates) #taking it to be the newest customer
for i in dates:
    delta = d1 - i
    days.append(delta)
data["Customer_For"] = days
data["Customer_For"] = pd.to_numeric(data["Customer_For"], errors="coerce")
```

In [15]:

```
print("Total categories in the feature Marital_Status:\n", data["Marital_Status"].value_
print("Total categories in the feature Education:\n", data["Education"].value_counts())
```

```
Total categories in the feature Marital_Status:
 Married     83
Together    48
Single      40
Divorced    23
Widow        4
Alone        2
Name: Marital_Status, dtype: int64

Total categories in the feature Education:
 Graduation    111
PhD            49
Master         35
Basic           5
Name: Education, dtype: int64
```

In [16]:

```
#Feature Engineering
#Age of customer today
data["Age"] = 2021-data["Year_Birth"]

#Total spendings on various items
data["Spent"] = data["MntWines"]+ data["MntFruits"]+ data["MntMeatProducts"]+ data["MntF

#Deriving living situation by marital status"Alone"
data["Living_With"]=data["Marital_Status"].replace({"Married":"Partner", "Together":"Par

#Feature indicating total children living in the household
data["Children"]=data["Kidhome"]+data["Teenhome"]

#Feature for total members in the householde
data["Family_Size"] = data["Living_With"].replace({"Alone": 1, "Partner":2})+ data["Chil

#Feature pertaining parenthood
data["Is_Parent"] = np.where(data.Children> 0, 1, 0)

#Segmenting education levels in three groups
data["Education"]=data["Education"].replace({"Basic":"Undergraduate","2n Cycle":"Undergr

#For clarity
data=data.rename(columns={"MntWines": "Wines","MntFruits":"Fruits","MntMeatProducts":"Me

#Dropping some of the redundant features
to_drop = ["Marital_Status", "Dt_Customer", "Z_CostContact", "Z_Revenue", "Year_Birth",
data = data.drop(to_drop, axis=1)
```

In [17]:

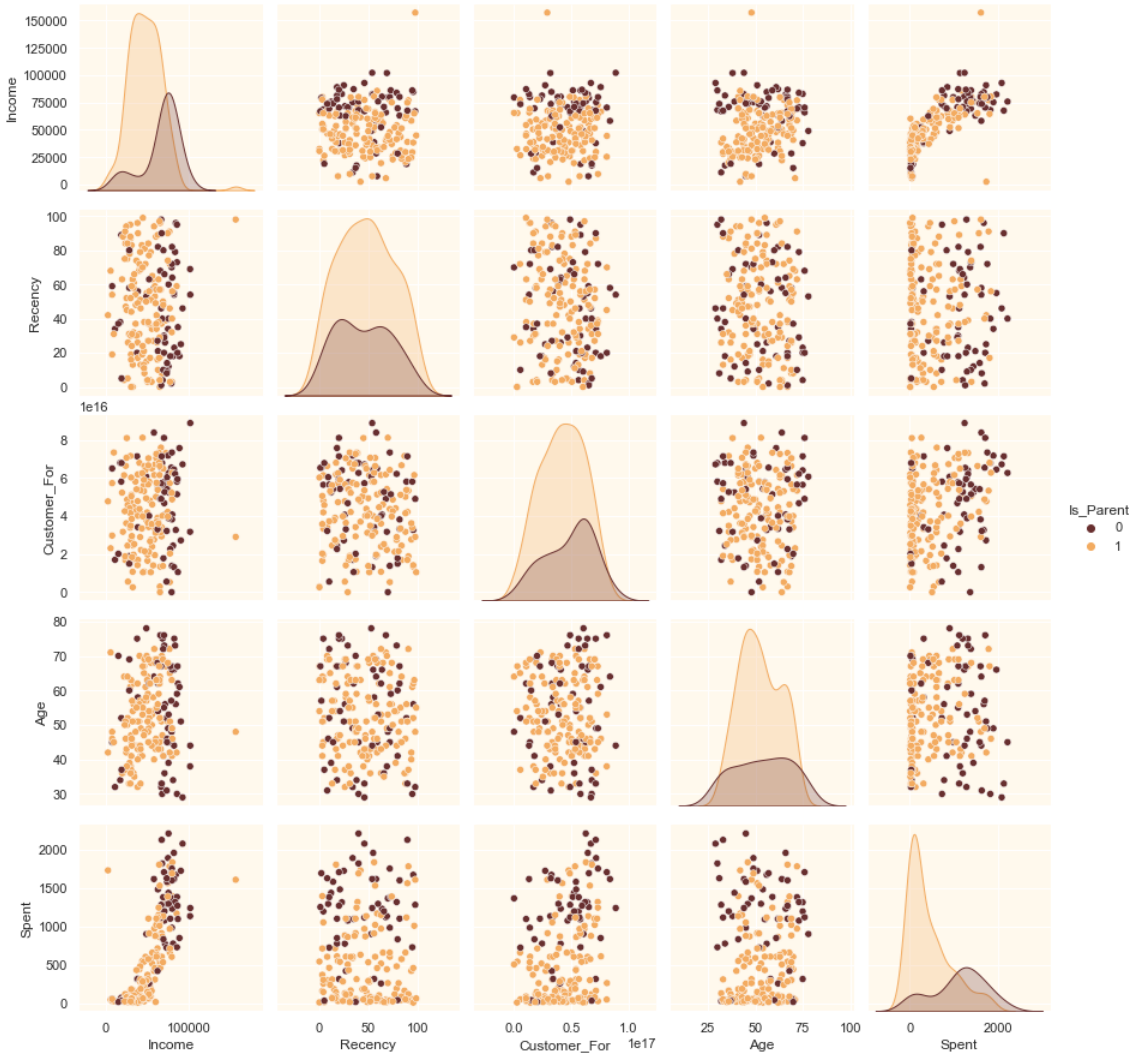
```
from matplotlib import colors
```

In [18]:

```
#To plot some selected features
#Setting up colors prefrences
sns.set(rc={"axes.facecolor":"#FFF9ED","figure.facecolor":"#FFF9ED"})
pallet = ["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#F3AB60"]
cmap = colors.ListedColormap(["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#F
#Plotting following features
To_Plot = [ "Income", "Recency", "Customer_For", "Age", "Spent", "Is_Parent"]
print("Reletive Plot Of Some Selected Features: A Data Subset")
plt.figure()
sns.pairplot(data[To_Plot], hue= "Is_Parent",palette= (["#682F2F","#F3AB60"]))
#Taking hue
plt.show()
```

```
Reletive Plot Of Some Selected Features: A Data Subset

<Figure size 432x288 with 0 Axes>
```



In [19]:

```
#Dropping the outliers by setting a cap on Age and income.
data = data[(data["Age"]<90)]
data = data[(data["Income"]<600000)]
print("The total number of data-points after removing the outliers are:", len(data))
```

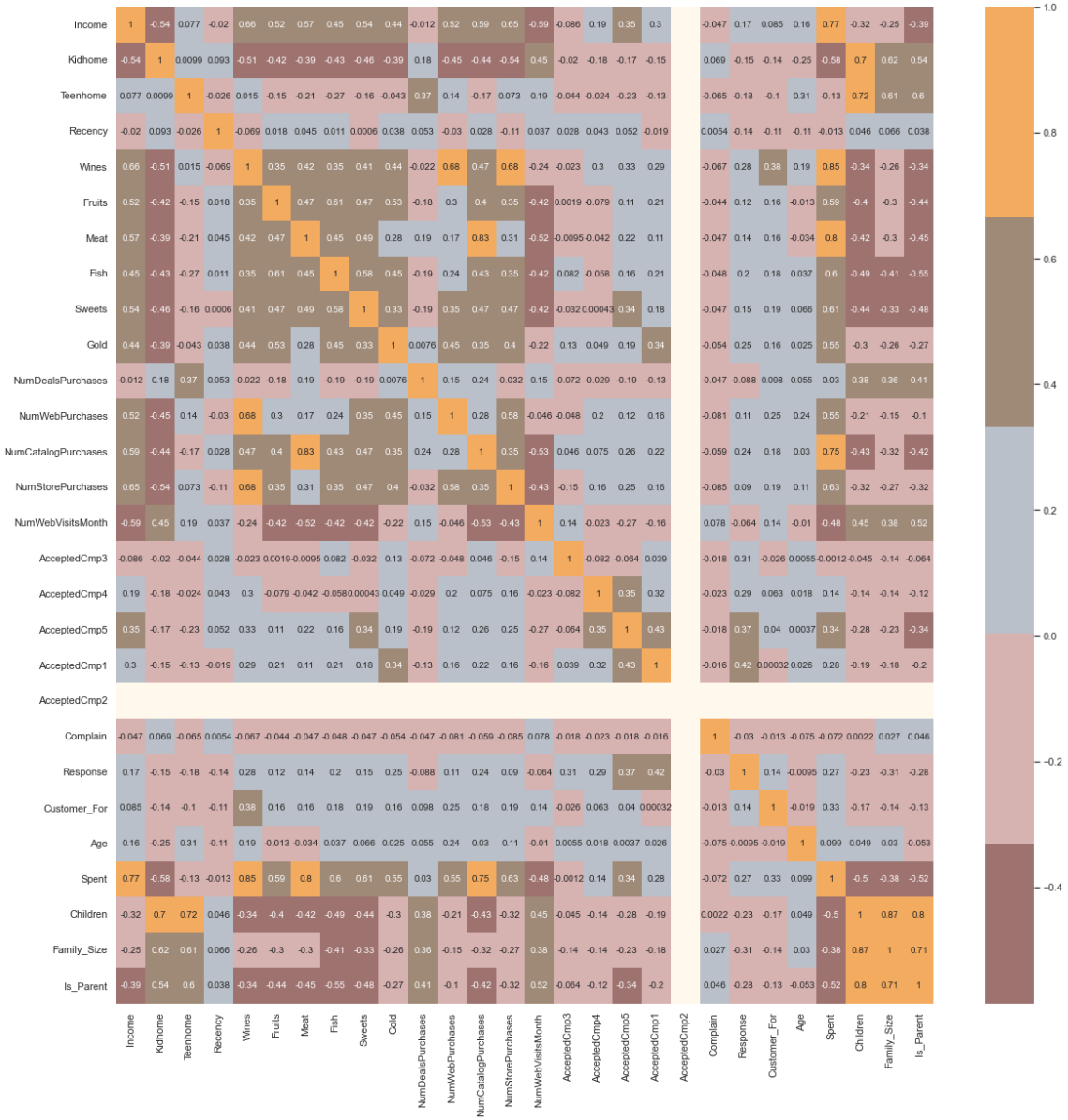
```
The total number of data-points after removing the outliers are: 200
```

In [20]:

```
#correlation matrix
corrmat= data.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corrmat,annot=True, cmap=cmap, center=0)
```

Out[20]:

```
<AxesSubplot: >
```



In [21]:

```
#Get list of categorical variables
s = (data.dtypes == 'object')
object_cols = list(s[s].index)
print("Categorical variables in the dataset:", object_cols)
```

```
Categorical variables in the dataset: ['Education', 'Living_With']
```

In [22]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt, numpy as np
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import AgglomerativeClustering
from matplotlib.colors import ListedColormap
from sklearn import metrics
import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
np.random.seed(42)
```

In [23]:

```
#Label Encoding the object dtypes.
LE=LabelEncoder()
for i in object_cols:
    data[i]=data[[i]].apply(LE.fit_transform)
print("All features are now numerical")
```

```
All features are now numerical
```

In [24]:

```
#Creating a copy of data
ds = data.copy()
# creating a subset of dataframe by dropping the features on deals accepted and promotio
cols_del = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1','AcceptedCmp2
ds = ds.drop(cols_del, axis=1)
#Scaling
scaler = StandardScaler()
scaler.fit(ds)
scaled_ds = pd.DataFrame(scaler.transform(ds),columns= ds.columns )
print("All features are now scaled")
```

```
All features are now scaled
```

In [25]:

```
#Scaled data to be used for reducing the dimensionality
print("Dataframe to be used for further modelling:")
scaled_ds.head()
```

Dataframe to be used for further modelling:

Out[25]:

	Education	Income	Kidhome	Teenhome	Recency	Wines	Fruits	Meat	
0	-0.859389	0.208993	-0.828325	-0.910720	0.399386	0.945144	1.551001	1.372401	2.
1	-0.859389	-0.311833	0.972381	0.857669	-0.317644	-0.933133	-0.615097	-0.671862	-0.
2	-0.859389	0.804053	-0.828325	-0.910720	-0.747862	0.316041	0.579991	-0.213796	1.
3	-0.859389	-1.181703	0.972381	-0.910720	-0.747862	-0.933133	-0.540404	-0.618863	-0.
4	0.969098	0.215838	0.972381	-0.910720	1.690040	-0.445504	0.430605	-0.247867	0.

5 rows × 23 columns

In [26]:

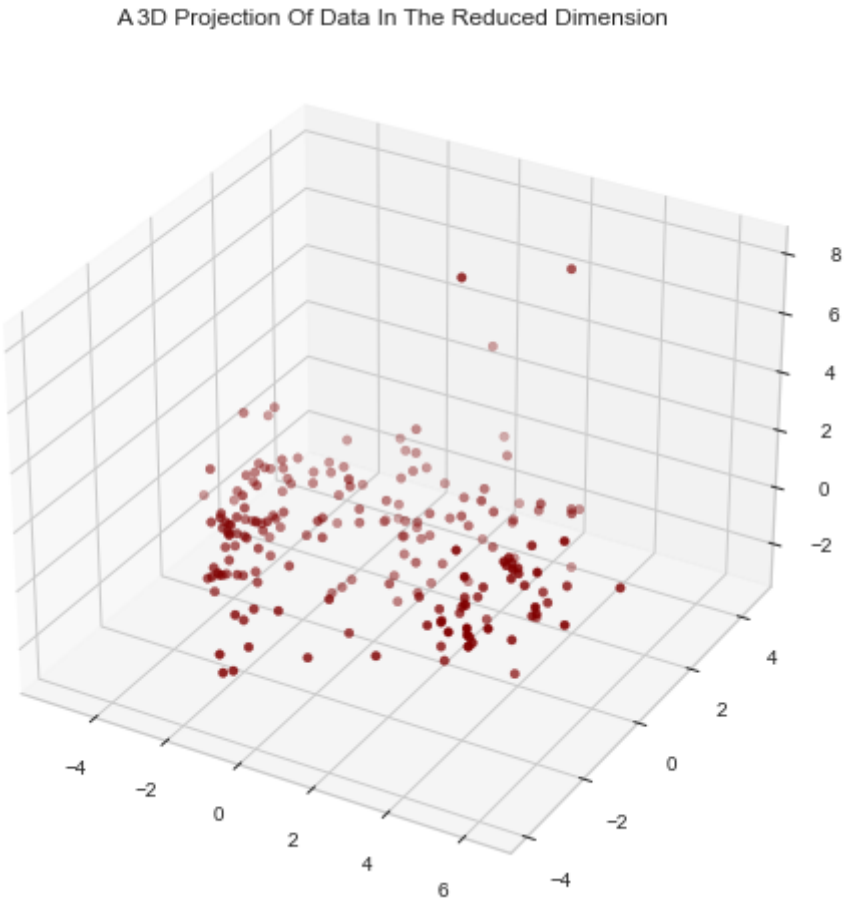
```
#Initiating PCA to reduce dimentions aka features to 3
pca = PCA(n_components=3)
pca.fit(scaled_ds)
PCA_ds = pd.DataFrame(pca.transform(scaled_ds), columns=(["col1","col2", "col3"]))
PCA_ds.describe().T
```

Out[26]:

	count	mean	std	min	25%	50%	75%	max
col1	200.0	1.154632e-16	2.852054	-5.251310	-2.549512	-0.788624	2.758482	6.389774
col2	200.0	-3.552714e-17	1.715182	-3.930605	-1.321979	-0.190908	1.324170	4.583010
col3	200.0	4.107825e-17	1.313574	-2.745802	-0.719173	-0.017188	0.603005	8.107841

In [27]:

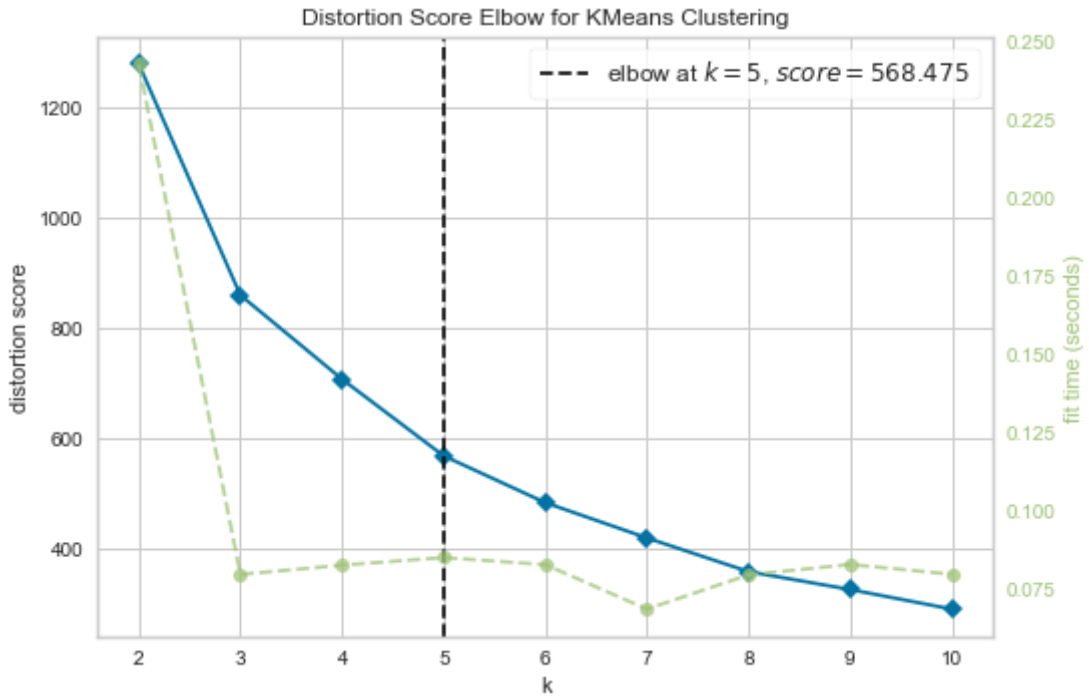
```
#A 3D Projection Of Data In The Reduced Dimension
x =PCA_ds["col1"]
y =PCA_ds["col2"]
z =PCA_ds["col3"]
#To plot
fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111, projection="3d")
ax.scatter(x,y,z, c="maroon", marker="o" )
ax.set_title("A 3D Projection Of Data In The Reduced Dimension")
plt.show()
```



In [28]:

```
# Quick examination of elbow method to find numbers of clusters to make.
print('Elbow Method to determine the number of clusters to be formed:')
Elbow_M = KElbowVisualizer(KMeans(), k=10)
Elbow_M.fit(PCA_ds)
Elbow_M.show()
```

```
Elbow Method to determine the number of clusters to be formed:
```



Out[28]:

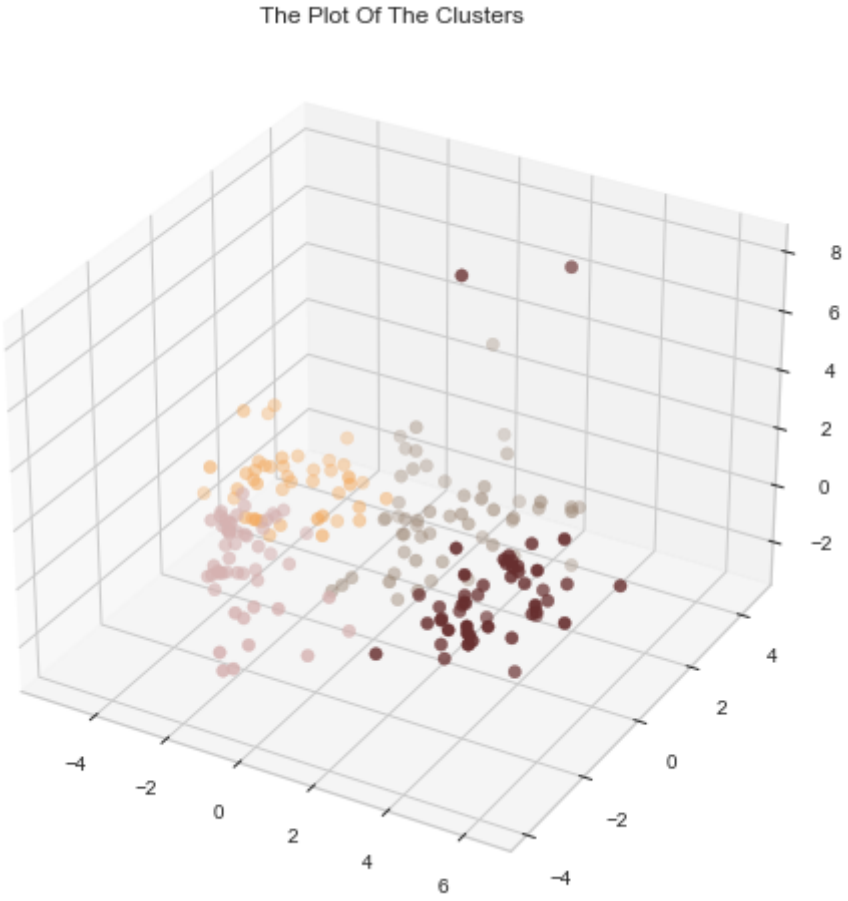
```
<AxesSubplot: title={'center': 'Distortion Score Elbow for KMeans Clusteri
ng'}, xlabel='k', ylabel='distortion score'>
```

In [29]:

```
#Initiating the Agglomerative Clustering model
AC = AgglomerativeClustering(n_clusters=4)
# fit model and predict clusters
yhat_AC = AC.fit_predict(PCA_ds)
PCA_ds["Clusters"] = yhat_AC
#Adding the Clusters feature to the orignal dataframe.
data["Clusters"]= yhat_AC
```

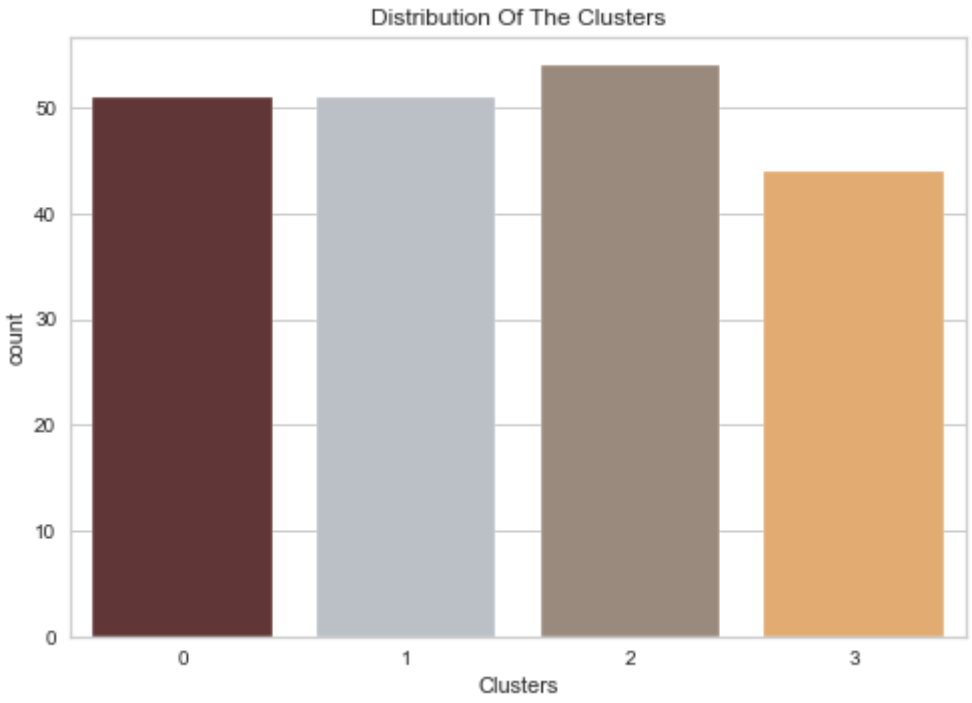
In [30]:

```
#Plotting the clusters
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds["Clusters"], marker='o', cmap = cmap )
ax.set_title("The Plot Of The Clusters")
plt.show()
```



In [31]:

```
#Plotting countplot of clusters
pal = ["#682F2F","#B9C0C9", "#9F8A78","#F3AB60"]
pl = sns.countplot(x=data["Clusters"], palette= pal)
pl.set_title("Distribution Of The Clusters")
plt.show()
```

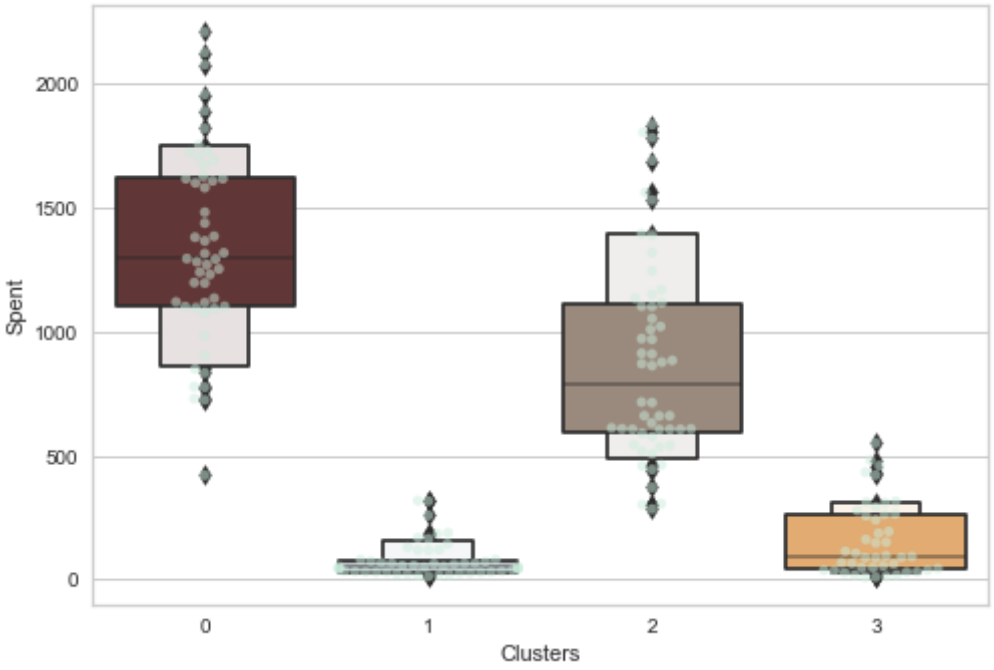


In [32]:

```
pl = sns.scatterplot(data = data,x=data["Spent"], y=data["Income"],hue=data["Clusters"],
pl.set_title("Cluster's Profile Based On Income And Spending")
plt.legend()
plt.show()
```

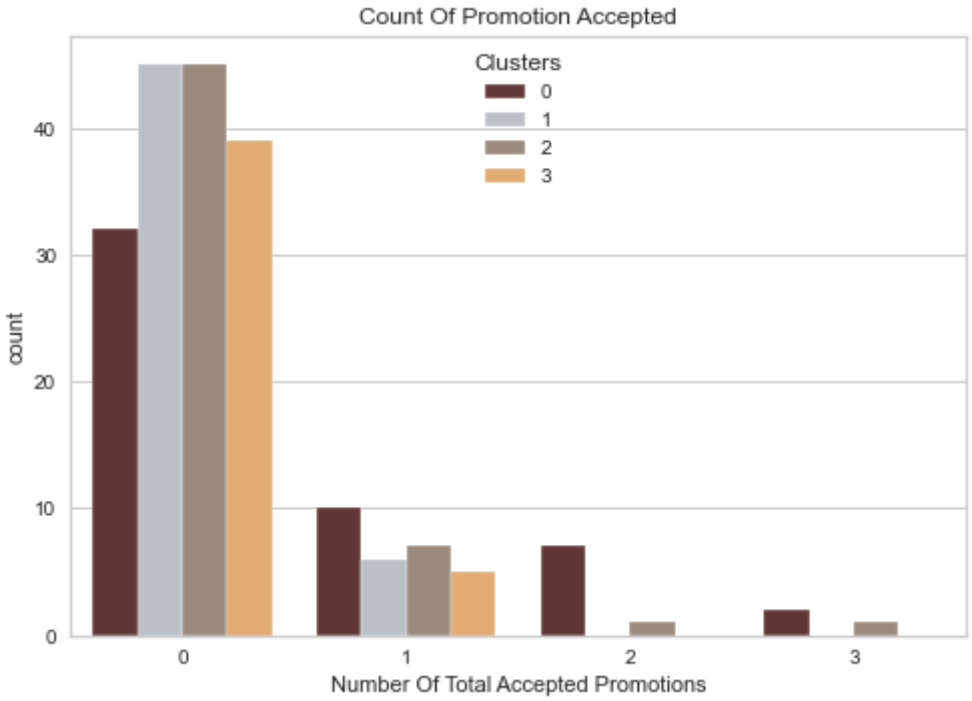
In [33]:

```
plt.figure()
pl=sns.swarmplot(x=data["Clusters"], y=data["Spent"], color= "#CBEDDD", alpha=0.5 )
pl=sns.boxenplot(x=data["Clusters"], y=data["Spent"], palette=pal)
plt.show()
```



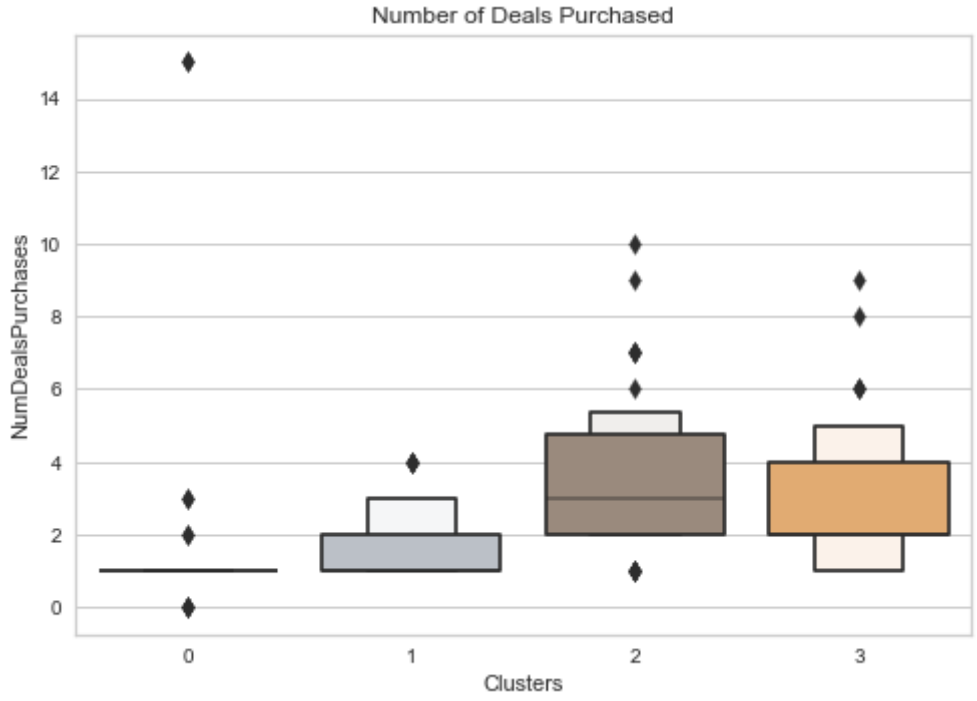
In [34]:

```
#Creating a feature to get a sum of accepted promotions
data["Total_Promos"] = data["AcceptedCmp1"]+ data["AcceptedCmp2"]+ data["AcceptedCmp3"]+
#Plotting count of total campaign accepted.
plt.figure()
pl = sns.countplot(x=data["Total_Promos"],hue=data["Clusters"], palette= pal)
pl.set_title("Count Of Promotion Accepted")
pl.set_xlabel("Number Of Total Accepted Promotions")
plt.show()
```



In [35]:

```
#Plotting the number of deals purchased
plt.figure()
pl=sns.boxenplot(y=data["NumDealsPurchases"],x=data["Clusters"], palette= pal)
pl.set_title("Number of Deals Purchased")
plt.show()
```



In [36]:

```
Personal = [ "Kidhome","Teenhome","Customer_For", "Age", "Children", "Family_Size", "Is_

for i in Personal:
    plt.figure()
    sns.jointplot(x=data[i], y=data["Spent"], hue =data["Clusters"], kind="kde", palette
    plt.show()
```

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