Final Project : Customer Personality Analysis

1. Problem Statement

Customer Personality Analysis is a detailed analysis of a company's ideal customers. It helps a business to better understand its customers and makes it easier for them to modify products according to the specific needs, behaviors and concerns of different types of customers.

Customer personality 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.

2. Main Objective of the Analysis

In this project, I will be performing an unsupervised clustering of data on the customer's records from a groceries firm's database. Customer segmentation is the practice of separating customers into groups that reflect similarities among customers in each cluster.

I will divide customers into segments to optimize the significance of each customer to the business. To modify products according to distinct needs and behaviours of the customers. It also helps the business to cater to the concerns of different types of customers.

3. Data Description

The dataset used in this analysis consists of 2240 datapoints and 29 attributes. It can be categorised into the following subsets :

People : Customer's Information

- 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

Read the first 5 rows in the dataset :

In [3]:

```
#Loading the dataset
data = pd.read_csv("../input/customer-personality-analysis/marketing_campaign.csv", sep
="\t")
print("Number of datapoints:", len(data))
data.head()
```

```
Number of datapoints: 2240
```

Out[3]:

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

5 rows × 29 columns

Acti

4. Data Cleaning

Data type and checking null in dataset :

In [4]:

```python
#Information on features
data.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  
```


```
 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
```

=> From the above output, we can conclude and note that:

- ***There are missing values in income***
- ***Dt_Customer that indicates the date a customer joined the database is not parsed as DateTime***
- ***There are some categorical features in our data frame; as there are some features in dtype: object). So we will need to encode them into numeric forms later.***

First of all, for the missing values, I am simply going to drop the rows that have missing income values.

In [5]:

```
#To remove the NA values
data = data.dropna()
print("The total number of data-points after removing the rows with missing values are:", l
en(data))
```

```
The total number of data-points after removing the rows with missing values are: 2216
```

In the next step, create a feature out of **"Dt_Customer"** that indicates the number of days a customer is registered in the firm's database. However, in order to keep it simple, taking this value relative to the most recent customer in the record.

In [6]:

```
data["Dt_Customer"] = pd.to_datetime(data["Dt_Customer"])
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: 2014-12-06
The oldest customer's enrolment date in the records: 2012-01-08
```

Creating a feature **("Customer_For")** of the number of days the customers started to shop in the store relative to the last recorded date.

In [7]:

```
#Created a feature "Customer_For"
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")
```

Now we will be exploring the unique values in the categorical features to get a clear idea of the data.

In [8]:

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

```
Total categories in the feature Marital_Status:
 Married      857
Together     573
Single       471
Divorced     232
Widow         76
Alone          3
Absurd         2
YOLO           2
Name: Marital_Status, dtype: int64

Total categories in the feature Education:
 Graduation    1116
PhD            481
Master         365
2n Cycle       200
Basic           54
Name: Education, dtype: int64
```

4. Features engeneering

- Extract the **"Age"** of a customer by the **"Year_Birth"** indicating the birth year of the respective person.
- Create another feature **"Spent"** indicating the total amount spent by the customer in various categories over the span of two years.
- Create another feature **"Living_With"** out of **"Marital_Status"** to extract the living situation of couples.
- Create a feature **"Children"** to indicate total children in a household that is, kids and teenagers.
- To get further clarity of household, Creating feature indicating **"Family_Size"**
- Create a feature **"Is_Parent"** to indicate parenthood status
- Lastly, I will create three categories in the **"Education"** by simplifying its value counts.
- Dropping some of the redundant features

In [9]:

```
#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["MntFishProducts"]+ data["MntSweetProducts"]+ data["MntGoldProds"]

#Deriving living situation by marital status"Alone"
data["Living_With"]=data["Marital_Status"].replace({"Married":"Partner", "Together":"Partner", "Absurd":"Alone", "Widow":"Alone", "YOLO":"Alone", "Divorced":"Alone", "Single":"Alone",})

#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["Children"]
```

Now that we have some new features let's have a look at the data's stats.

In [10]:

```
data.describe()
```

Out[10]:

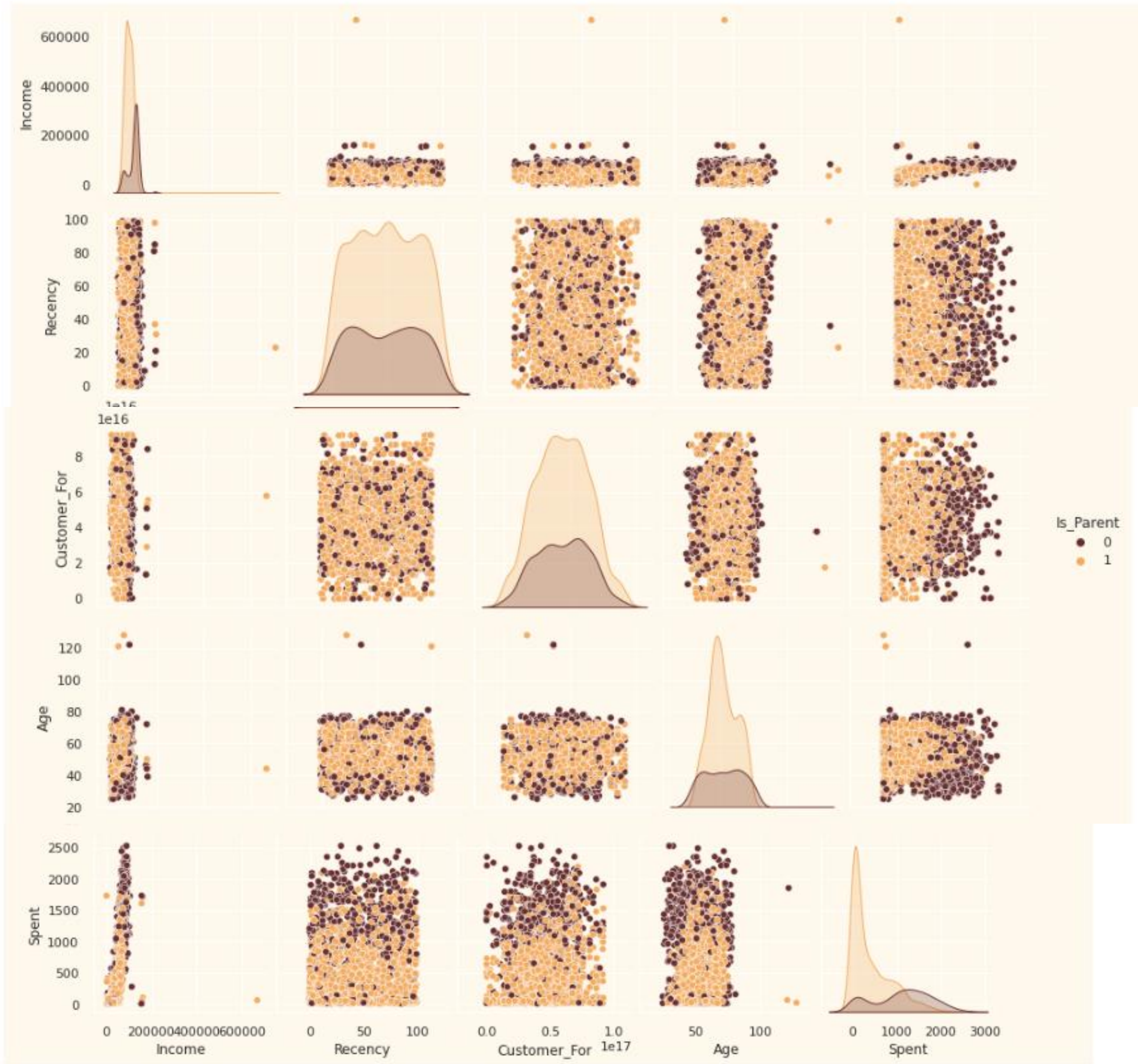
	Income	Kidhome	Teenhome	Recency	Wines	Fruits	Meat	Fish	Sweets	Gold
count	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.00
mean	52247.251354	0.441787	0.505415	49.012635	305.091606	26.356047	166.995939	37.637635	27.028881	43.9652
std	25173.076661	0.536896	0.544181	28.948352	337.327920	39.793917	224.283273	54.752082	41.072046	51.8154
min	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	35303.000000	0.000000	0.000000	24.000000	24.000000	2.000000	16.000000	3.000000	1.000000	9.00000
50%	51381.500000	0.000000	0.000000	49.000000	174.500000	8.000000	68.000000	12.000000	8.000000	24.5000
75%	68522.000000	1.000000	1.000000	74.000000	505.000000	33.000000	232.250000	50.000000	33.000000	56.0000
max	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	262.000000	321.000

Let’s take a look at the broader view of the data. I will plot some of the selected features.

In [11]:

```
#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", "#F3AB60"])
#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()
```

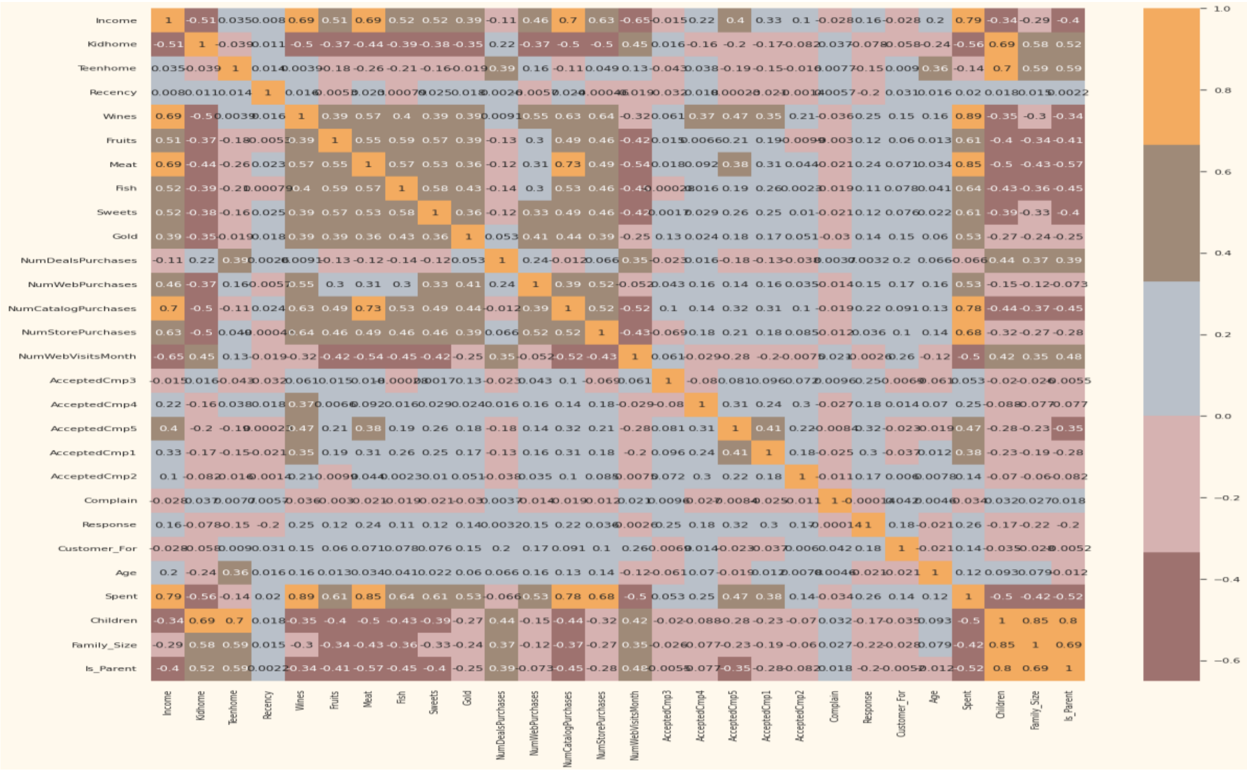
<Figure size 576x396 with 0 Axes>



⇨ ***Clearly, there are a few outliers in the Income and Age features. I will be deleting the outliers in the data.***

Next, let's look at the correlation amongst the features. (Excluding the categorical attributes at this point).

In [13]:
```
#correlation matrix
corrmat= data.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corrmat,annot=True, cmap=cmap, center=0)
```



The data is quite clean and the new features have been included. I will proceed to the next step. That is, preprocessing the data.

5. Data Preprocessing

Preprocessing the data to perform clustering operations. **The following steps are applied to preprocess the data:**

- Label encoding the categorical features
- Scaling the features using the standard scaler
- Creating a subset dataframe for dimensionality reduction

In [14]:

```
#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 [15]:

```
#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 [16]:

```python
#Creating a copy of data
ds = data.copy()
# creating a subset of dataframe by dropping the features on deals accepted and promotions
cols_del = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1','AcceptedCmp2', 'Complain', 'Response']
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
```

```
Dataframe to be used for further modelling:
```

Out[17]:

	Education	Income	Kidhome	Teenhome	Recency	Wines	Fruits	Meat	Fish	Sweets	...	NumCatalogPurchases
0	-0.893586	0.287105	-0.822754	-0.929699	0.310353	0.977660	1.552041	1.690293	2.453472	1.483713	...	2.503607
1	-0.893586	-0.260882	1.040021	0.908097	-0.380813	-0.872618	-0.637461	-0.718230	-0.651004	-0.634019	...	-0.571340
2	-0.893586	0.913196	-0.822754	-0.929699	-0.795514	0.357935	0.570540	-0.178542	1.339513	-0.147184	...	-0.229679
3	-0.893586	-1.176114	1.040021	-0.929699	-0.795514	-0.872618	-0.561961	-0.655787	-0.504911	-0.585335	...	-0.913000
4	0.571657	0.294307	1.040021	-0.929699	1.554453	-0.392257	0.419540	-0.218684	0.152508	-0.001133	...	0.111982

6. Summary of Data Exploration and Cleaning

In our analysis above, we have concluded following:

- Senior customers tend to buy more wines
- Customers with higher incomes tend to buy more as compared to middle and low incomes. But the senior customer's tend to buy more.
- Absurd and widowed customers tend to buy more than others, these two categories of customers based on their marital status buy more wines.
- Customers have no children buy more as compared to those who have more children.
- Customers with low incomes tend to visit web more frequently and most of the customer tend to make purchases by visit the store.
- The regression analysis shows that there is positive relationship between number of webvisits and the purchases they make.

7. DIMENSIONALITY REDUCTION

In this problem, there are many factors on the basis of which the final classification will be done. These factors are basically attributes or features. The higher the number of features, the harder it is to work with it. Many of these features are correlated, and hence redundant. This is why I will be performing dimensionality reduction on the selected features before putting them through a classifier. *Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables.*

Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss.

Steps in this section:

- Dimensionality reduction with PCA
- Plotting the reduced dataframe

Dimensionality reduction with PCA
For this project, I will be reducing the dimensions to 3.

In [18]:

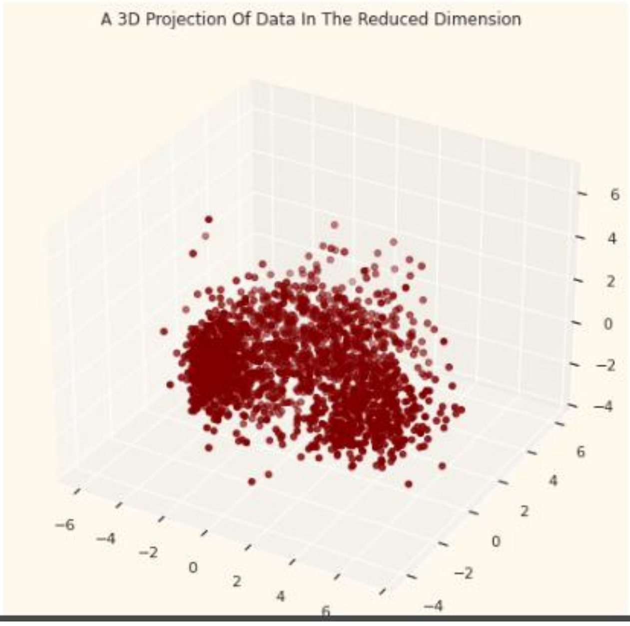
```
#Initiating PCA to reduce dimentions aka features to 3
pca = PCA(n_components=3)
pca.fit(scaled_ds)
PCA_ds = pd.DataFrame(pca.transform(scaled_ds), columns=(["col1","col2", "col3"]))
PCA_ds.describe().T
```

Out[18]:

	count	mean	std	min	25%	50%	75%	max
col1	2212.0	-1.116246e-16	2.878377	-5.969394	-2.538494	-0.780421	2.383290	7.444305
col2	2212.0	1.105204e-16	1.706839	-4.312196	-1.328316	-0.158123	1.242289	6.142721
col3	2212.0	3.049098e-17	1.221956	-3.530416	-0.829067	-0.022692	0.799895	6.611222

In [19]:

```
#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()
```



8. Clustering

Now that I have reduced the attributes to three dimensions, I will be performing clustering via Agglomerative clustering. Agglomerative clustering is a hierarchical clustering method. It involves merging examples until the desired number of clusters is achieved.

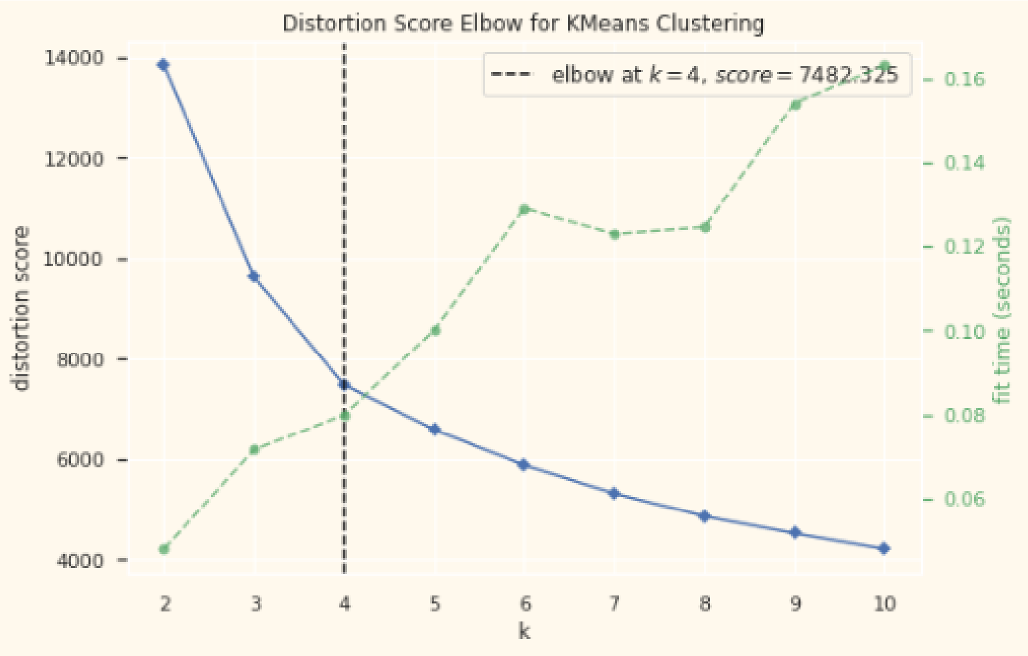
Steps involved in the Clustering

- Elbow Method to determine the number of clusters to be formed
- Clustering via Agglomerative Clustering
- Examining the clusters formed via scatter plot

In [20]:

```
# 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()
```

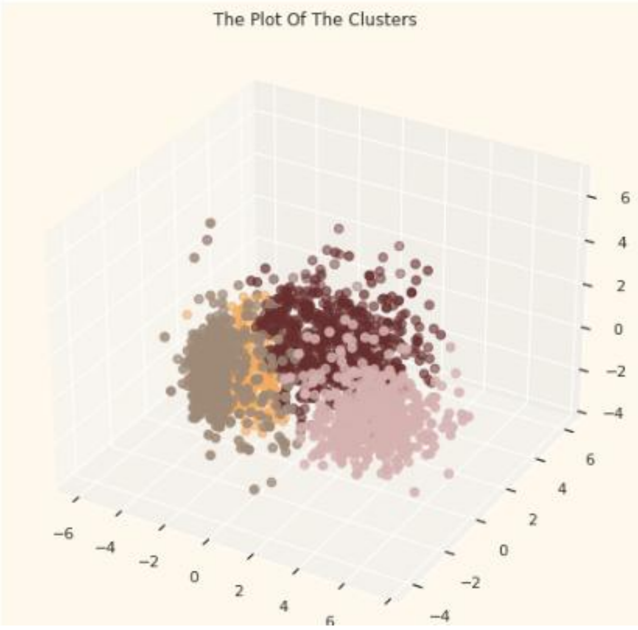
```
Elbow Method to determine the number of clusters to be formed:
```



The above cell indicates that four will be an optimal number of clusters for this data. Next, we will be fitting the Agglomerative Clustering Model to get the final clusters.

In [22]:

```
#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()
```



9. EVALUATING MODELS

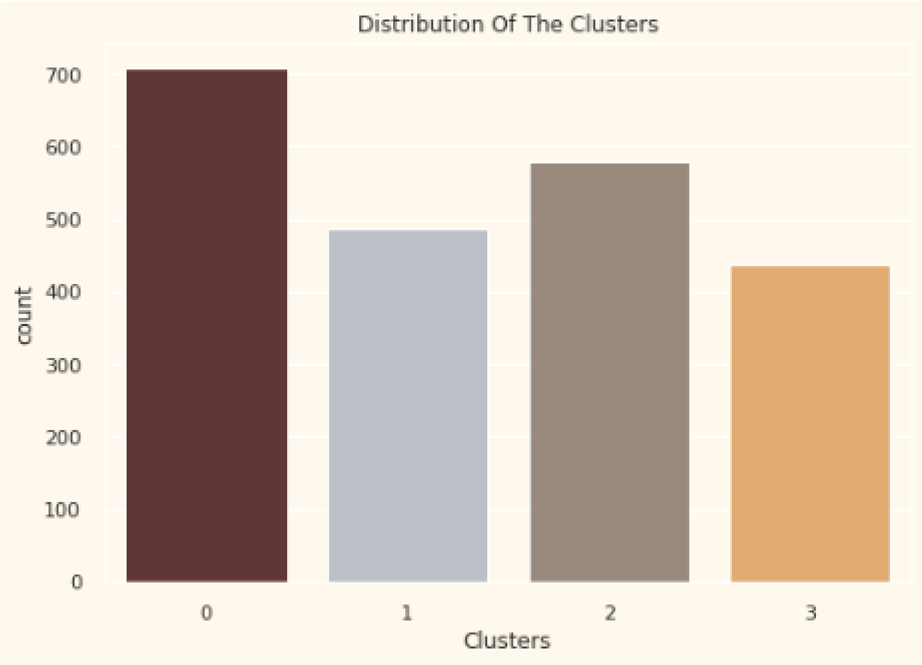
Since this is an unsupervised clustering. We do not have a tagged feature to evaluate or score our model. The purpose of this section is to study the patterns in the clusters formed and determine the nature of the clusters' patterns.

For that, we will be having a look at the data in light of clusters via exploratory data analysis and drawing conclusions.

Firstly, let us have a look at the group distribution of clustring

In [23]:

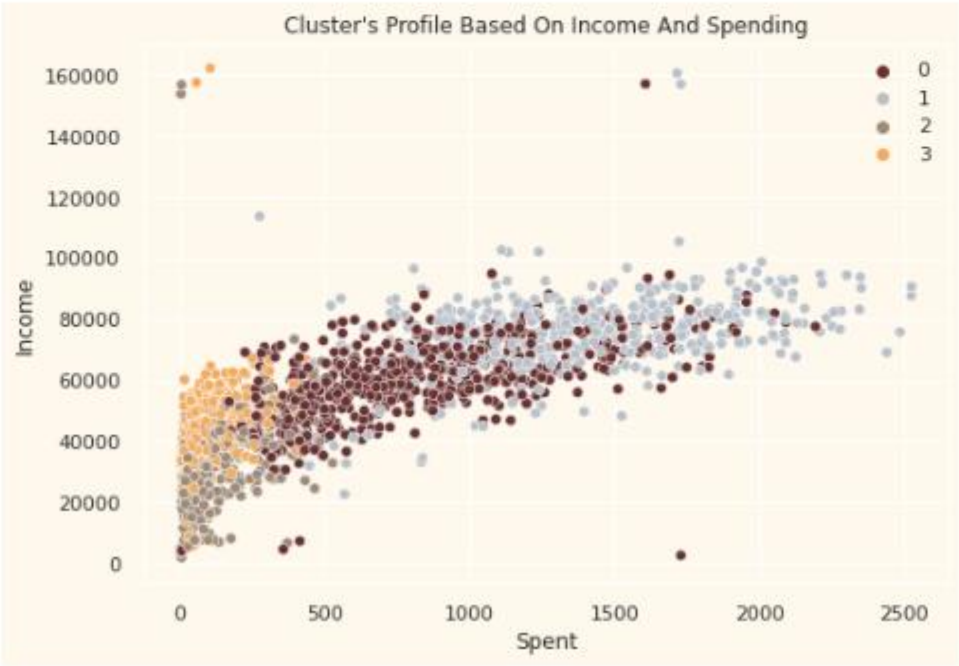
```
#Plotting countplot of clusters
pal = ["#682F2F","#B9C0C9", "#9F8A78","#F3AB60"]
pl = sns.countplot(x=data["Clusters"], palette= pal)
pl.set_title("Distribution Of The Clusters")
plt.show()
```



The clusters seem to be fairly distributed.

In [24]:

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



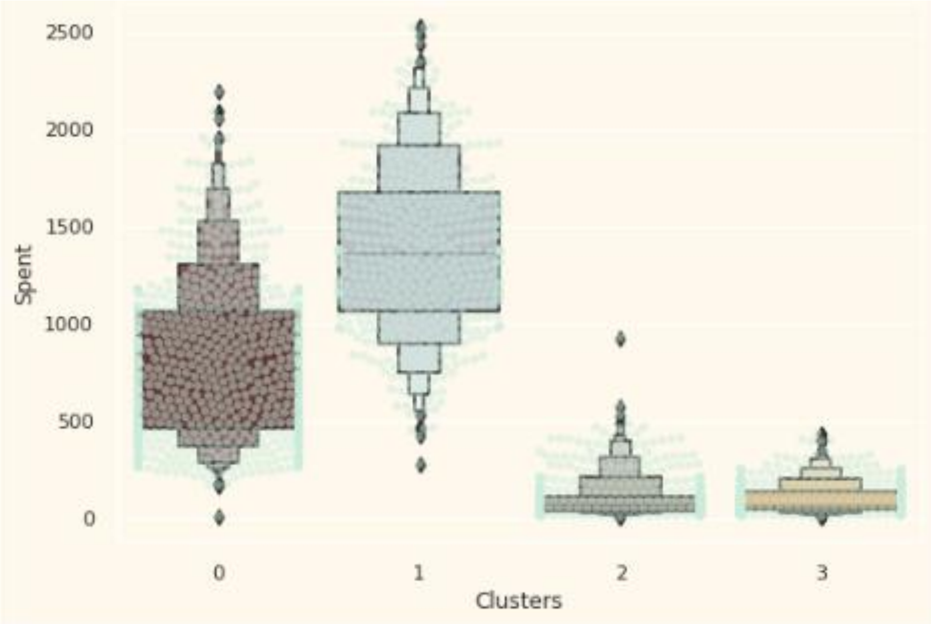
Income vs spending plot shows the clusters pattern

- group 0: high spending & average income
- group 1: high spending & high income
- group 2: low spending & low income
- group 3: high spending & low income

Next, looking at the detailed distribution of clusters as per the various products in the data. Namely: Wines, Fruits, Meat, Fish, Sweets and Gold

In [25]:

```
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()
```

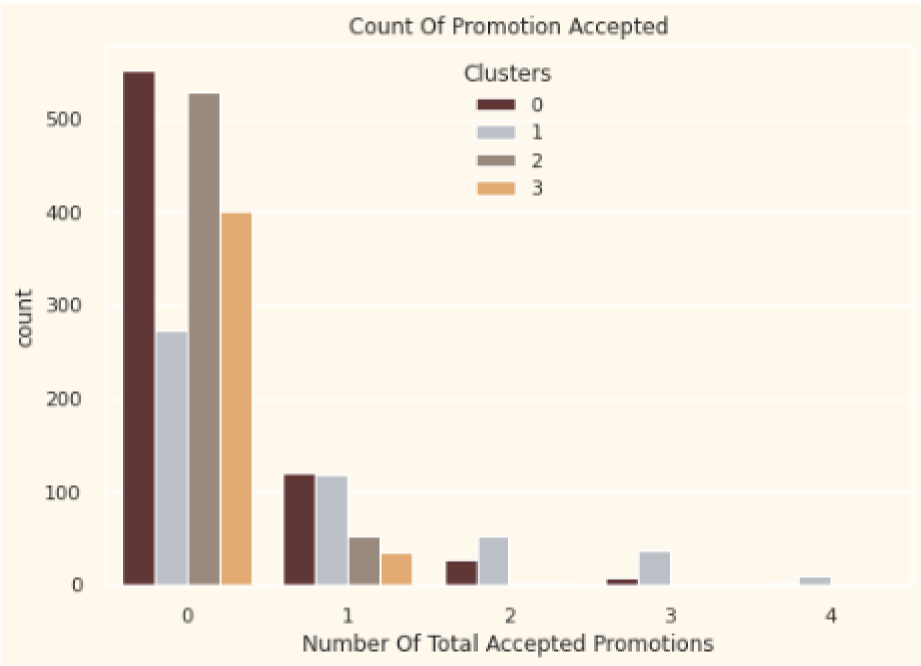


From the above plot, it can be clearly seen that cluster 1 is our biggest set of customers closely followed by cluster 0. We can explore what each cluster is spending on for the targeted marketing strategies.

Let's next explore how did our campaigns do in the past.

In [26]:

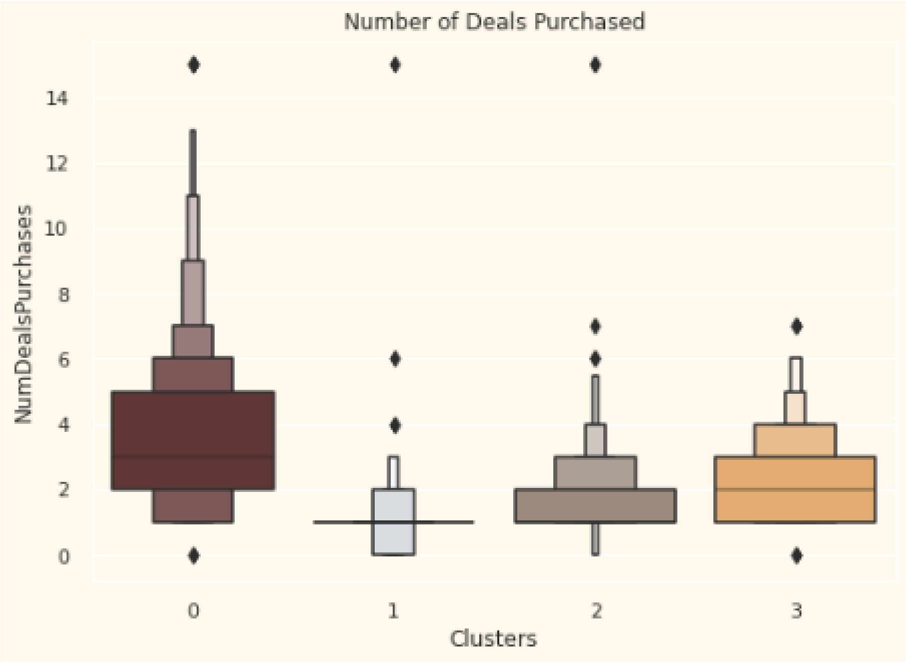
```
#Creating a feature to get a sum of accepted promotions
data["Total_Promos"] = data["AcceptedCmp1"]+ data["AcceptedCmp2"]+ data["AcceptedCmp3"]+ data["AcceptedCmp4"]+ data["AcceptedCmp5"]
#Plotting count of total campaign accepted.
plt.figure()
pl = sns.countplot(x=data["Total_Promos"],hue=data["Clusters"], palette= pal)
pl.set_title("Count Of Promotion Accepted")
pl.set_xlabel("Number Of Total Accepted Promotions")
plt.show()
```

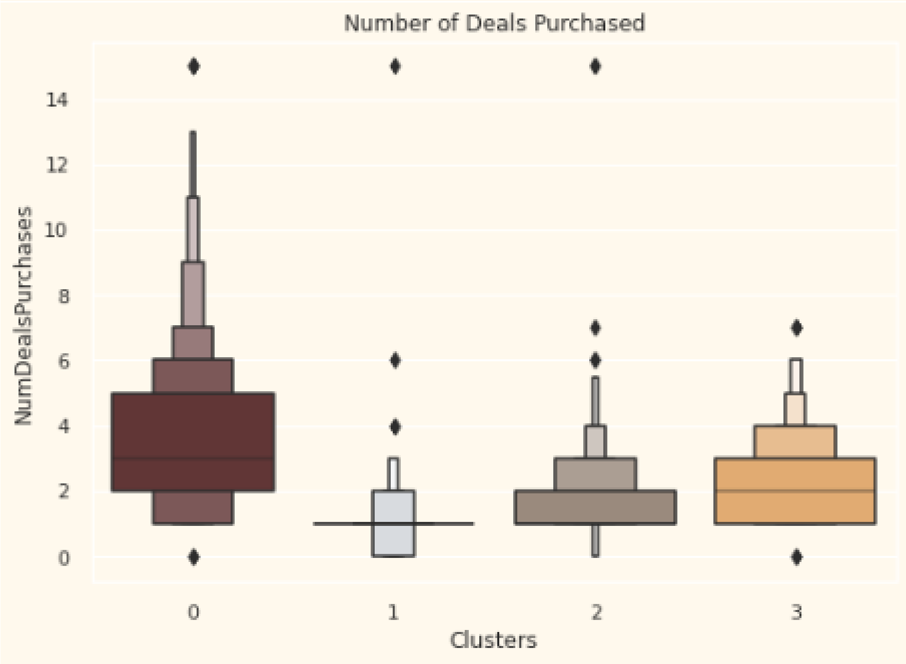


There has not been an overwhelming response to the campaigns so far. Very few participants overall. Moreover, no one part take in all 5 of them. Perhaps better-targeted and well-planned campaigns are required to boost sales.

In [27]:

```
#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()
```





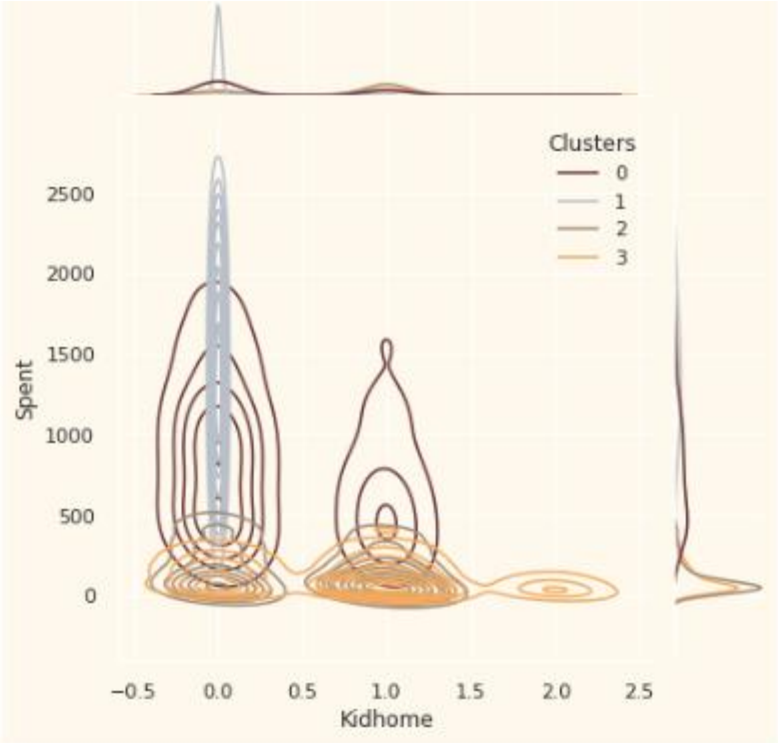
Unlike campaigns, the deals offered did well. It has best outcome with cluster 0 and cluster 3. However, our star customers cluster 1 are not much into the deals. Nothing seems to attract cluster 2 overwhelmingly.

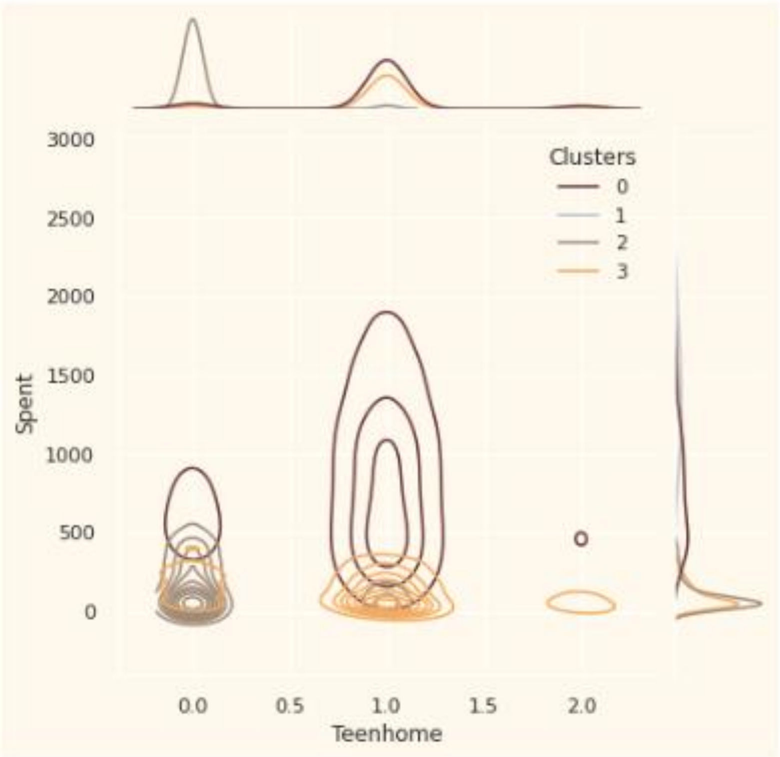
10. PROFILING

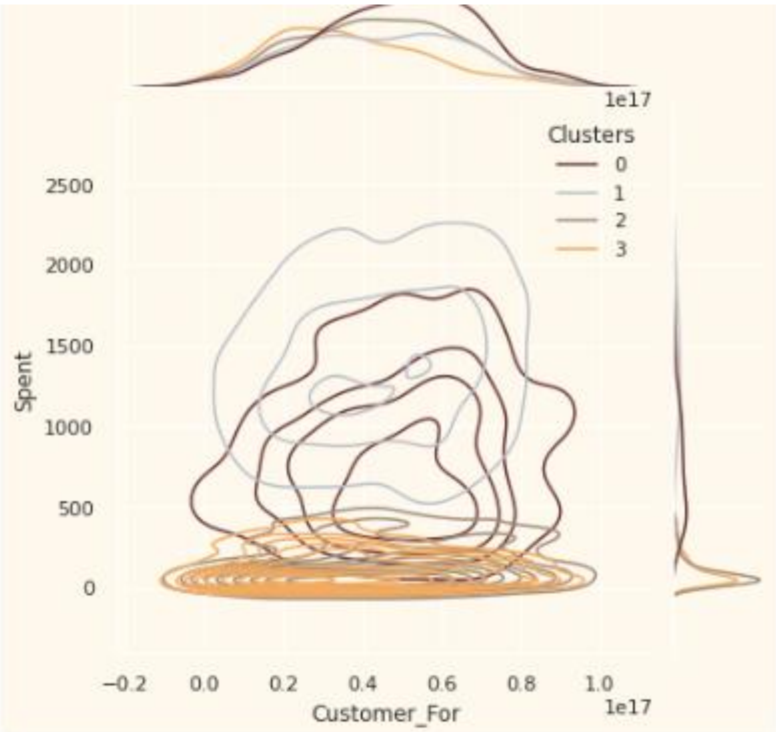
Now that we have formed the clusters and looked at their purchasing habits. Let us see who all are there in these clusters. For that, we will be profiling the clusters formed and come to a conclusion about who is our star customer and who needs more attention from the retail store's marketing team.

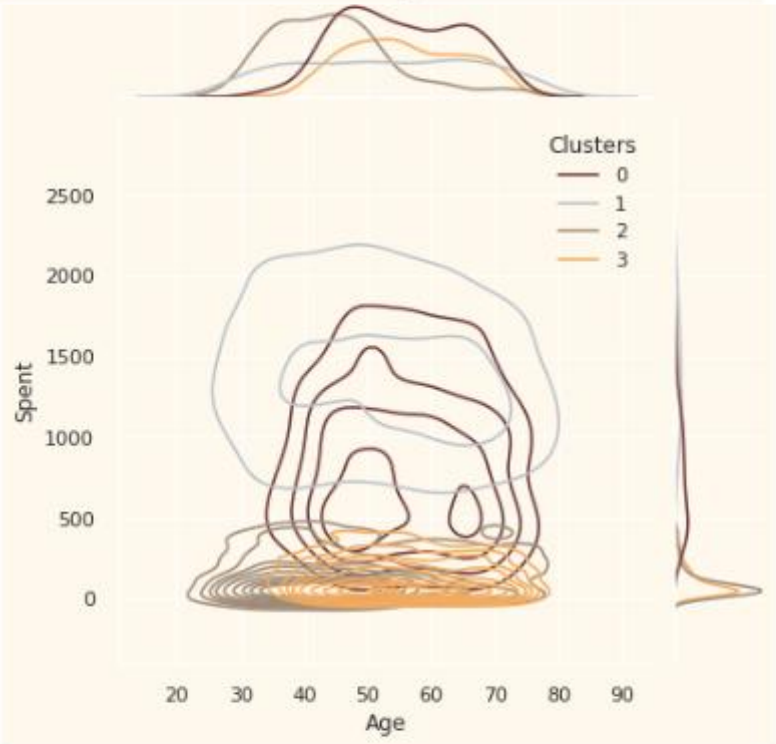
To decide that I will be plotting some of the features that are indicative of the customer's personal traits in light of the cluster they are in. On the basis of the outcomes, I will be arriving at the conclusions.

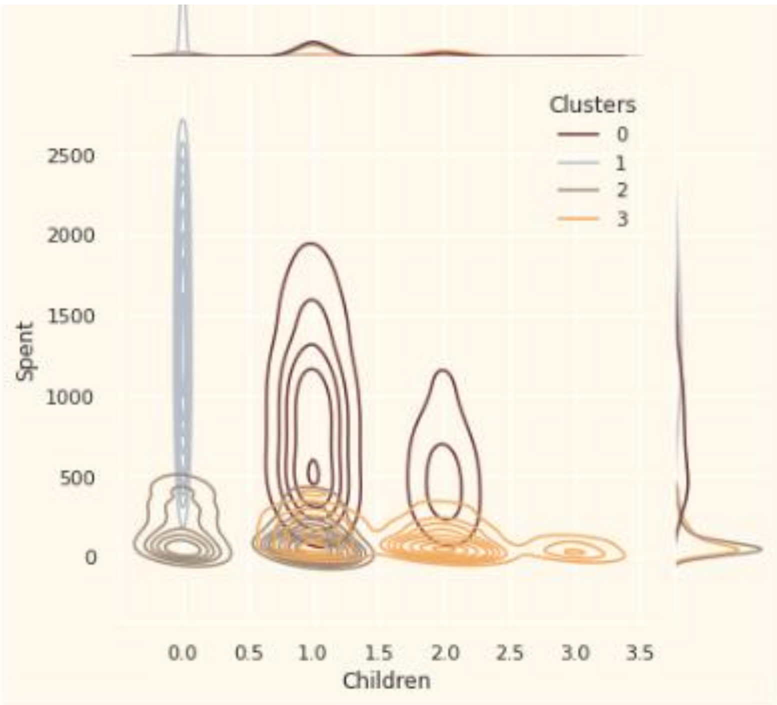
In [29]:
```
Personal = [ "Kidhome","Teenhome","Customer_For", "Age", "Children", "Family_Size", "Is_Parent", "Education","Living_With"]

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

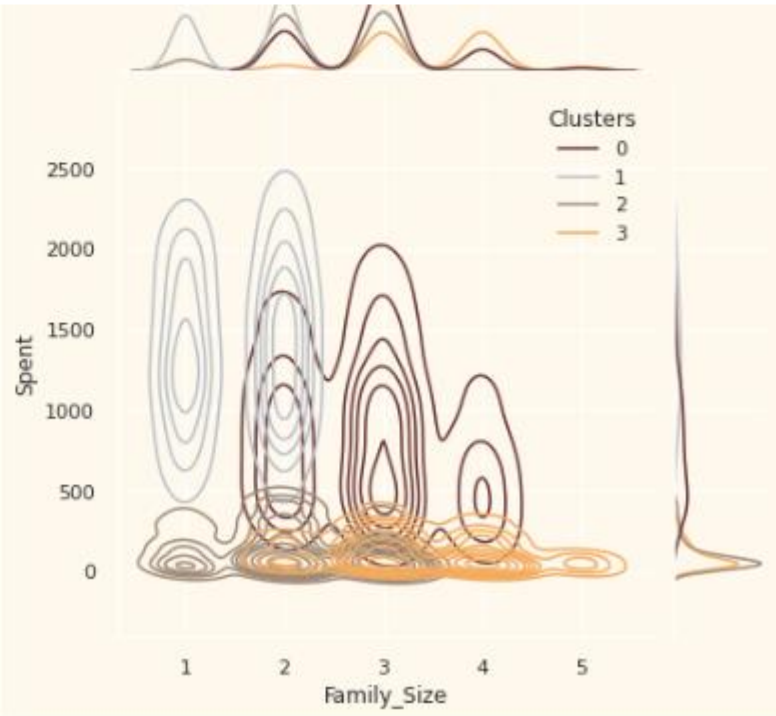


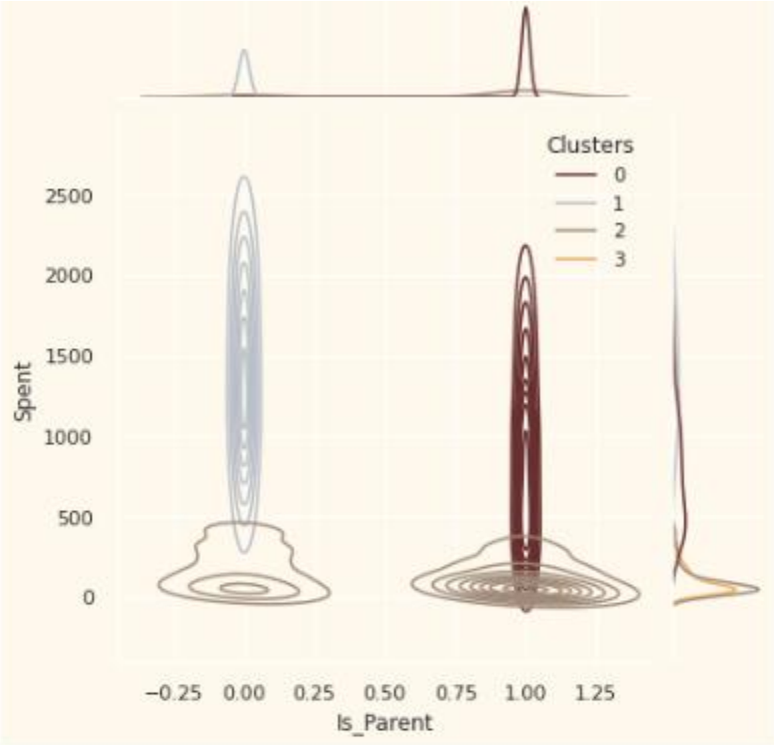


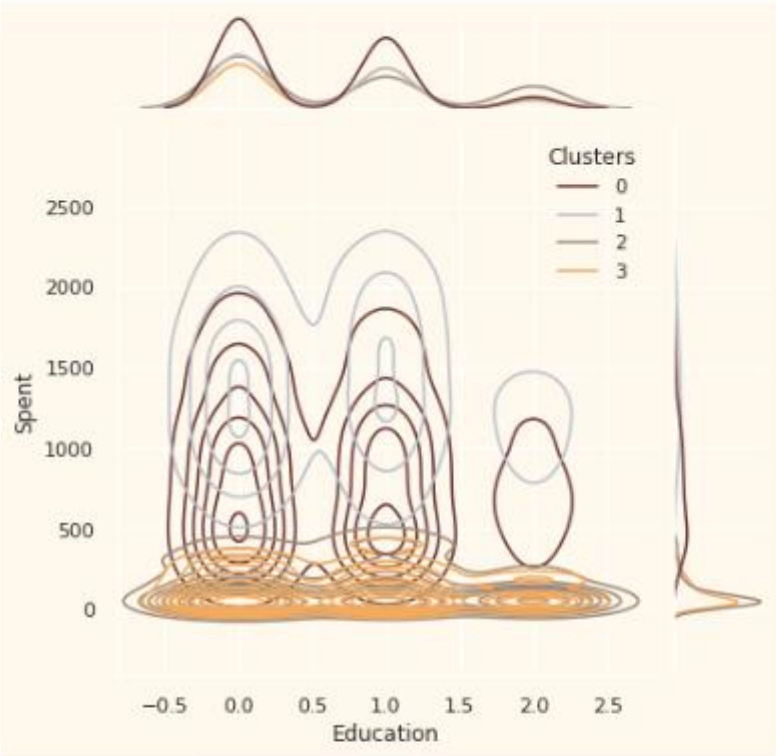

Clusters
0
1
2
3
Spent
2500
2000
1500
1000
500
0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1e17
Customer_For
1e17

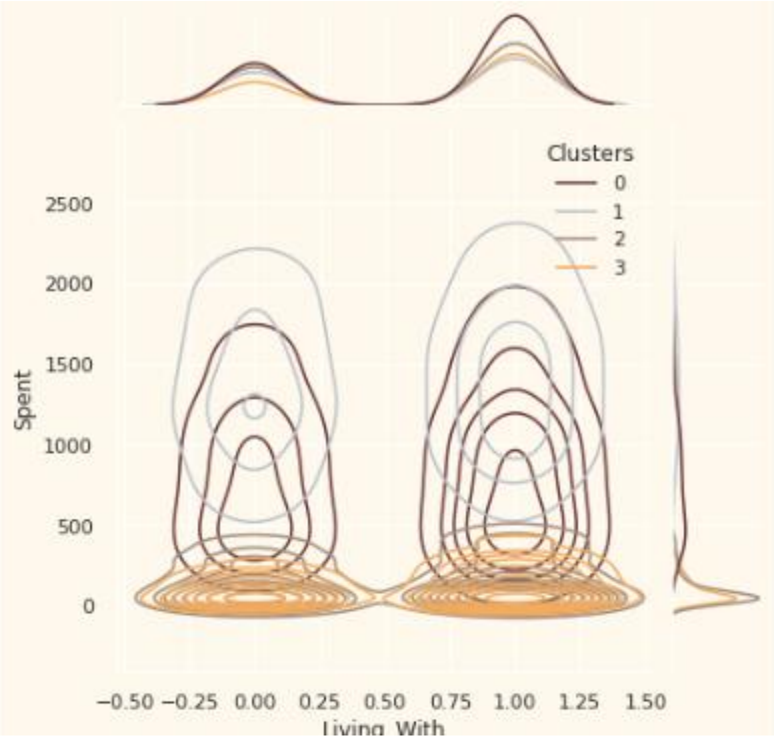

Clusters
0
1
2
3
Spent
2500
2000
1500
1000
500
0
20
30
40
50
60
70
80
90
Age


Clusters
0
1
2
3
2500
2000
1500
1000
500
0
Spent
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
Children


Clusters
0
1
2
3
2500
2000
1500
1000
500
0
Spent
1 2 3 4 5
Family_Size


Clusters
0
1
2
3
2500
2000
1500
1000
500
0
Spent
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
Is_Parent


Clusters
0
1
2
3
2500
2000
1500
1000
500
0
Spent
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
Education



11. Key Findings and Insights

Profiling the clusters :

- ***About Cluster Number : 0***
  - Are a definitely a parent
  - At the max have 4 members in the family and at least 2
  - Single parents ara s subset of this group
  - Most have a teenager at home
  - Relatively older

- ***About Cluster Number : 1***
  - Are a definetly not a parent
  - At the max are only 2 members in the family
  - At slight majority of couples over single peaple
  - Span all ages
  - A high income group

- ***About Cluster Number : 2***
  - The majority of these people are parents
  - At the max are 3 members in the family
  - They majorly have one kid (anod not teenagers, typically)
  - Relatively younger

- ***About Cluster Number : 3***
  - They are definitly a parent
  - At the max are 5 members in the family and at least 2
  - Relatively older
  - A lower-income group

In this project, we performed unsupervised clustering. We did use dimensionality reduction followed by agglomerative clustering. W came up with 4 clusters and further used them in profiling customers in clusters according to their family structures and income/spending. This can be used in planning better marketing strategies.

12. Model Flaws and Future Work

Model Flaws

1. **Cluster Interpretability**: The clusters formed might not have clear or distinct interpretations. Since the clustering is unsupervised, it may group customers in a way that is not intuitive or useful for marketing purposes.
2. **Dimensionality Reduction Limitations**: The process of dimensionality reduction (e.g., PCA) may result in the loss of important information, which could affect the quality and accuracy of the clusters.
3. **Agglomerative Clustering Scalability**: Agglomerative clustering has computational complexity issues, especially with large datasets. If your data grows, this could become a problem.
4. **Fixed Number of Clusters**: The decision to fix the number of clusters at 4 may not reflect the natural grouping within the data. This could lead to over-simplification or inappropriate grouping.
5. **Assumption of Homogeneity**: The model assumes homogeneity within each cluster, which might not be true, leading to potential misclassification or inaccurate profiling.
6. **Ignoring Temporal Dynamics**: If the data has any temporal aspect (e.g., spending habits over time), the static clustering approach might miss these dynamics.

Future Work :

Plan of Action: To address these limitations, the following steps are recommended:

1. **Cluster Validation and Refinement**: Employ techniques such as silhouette analysis or Davies-Bouldin index to validate the clusters and refine them if necessary. Consider using a different number of clusters to see if that offers better segmentation.
2. **Experiment with Different Algorithms**: Try other clustering algorithms such as DBSCAN, k-means, or Gaussian Mixture Models to see if they yield more meaningful clusters.
3. **Improve Dimensionality Reduction**: Explore non-linear dimensionality reduction techniques (e.g., t-SNE, UMAP) that might better capture the complex relationships in the data.
4. **Incorporate Domain Knowledge**: Integrate more domain-specific features into the model, such as customer behavior over time, to improve the clustering quality.
5. **Dynamic Clustering**: Explore dynamic or online clustering methods that can adjust as new data comes in, reflecting changes in customer behavior over time.
6. **A/B Testing**: Use A/B testing to evaluate the effectiveness of marketing strategies developed based on these clusters in real-world scenarios, allowing for data-driven refinements.