1) Importing Libraries

In [62]:
```
import numpy as np
import pandas as pd
import datetime
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import colors
import seaborn as sns
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 sklearn.cluster import SpectralClustering
from sklearn.ensemble import VotingClassifier
from scipy.spatial.distance import pdist, squareform
from scipy.cluster.hierarchy import linkage, fcluster
from sklearn.metrics import silhouette_samples
from sklearn.base import clone
from scipy.stats import mode
from sklearn.metrics import silhouette_score
from matplotlib.colors import ListedColormap
from sklearn import metrics
import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

2) Loading Data

In [63]:
```
data = pd.read_csv("C:/Users/Ravisankar/OneDrive/Desktop/Major project/marketing_ca
print("Number of datapoints:", len(data))
data.head()
```

```
Number of datapoints: 2240
```

Out[63]:

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

5 rows × 29 columns

In [64]:
```
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  
 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
```

3) Data Cleaning and feature Engineering

In [65]:
```
data = data.dropna()
print("The total number of data-points after removing the rows with missing values 
```

```
The total number of data-points after removing the rows with missing values are: 2
216
```

Feature Engineering: Creating new features out of given data

1) [Customer_For] from [Dt_Customer]: No of days customer has been

done shopping relative to newest record

2) [Age] from [Year_Birth]: Age of the customers

3) [spent] from [various products]: Total spending of customers

4) [living_with] from [martial_status]: So we divide 4 catogeries into 2[Partner, alone].

5) [children] from [kidhome]+[teenhome]: redundancy removing

In [66]:
```
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
```

In [67]:
```
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 [68]:
```
print("Total categories in the feature Marital_Status:\n", data["Marital_Status"].v
print("Total categories in the feature Education:\n", data["Education"].value_count
```

```
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
```

In [69]:
```
data["Age"] = 2021-data["Year_Birth"]

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

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

#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[

#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":"Un

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

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

In [70]:
```
data.describe()
```

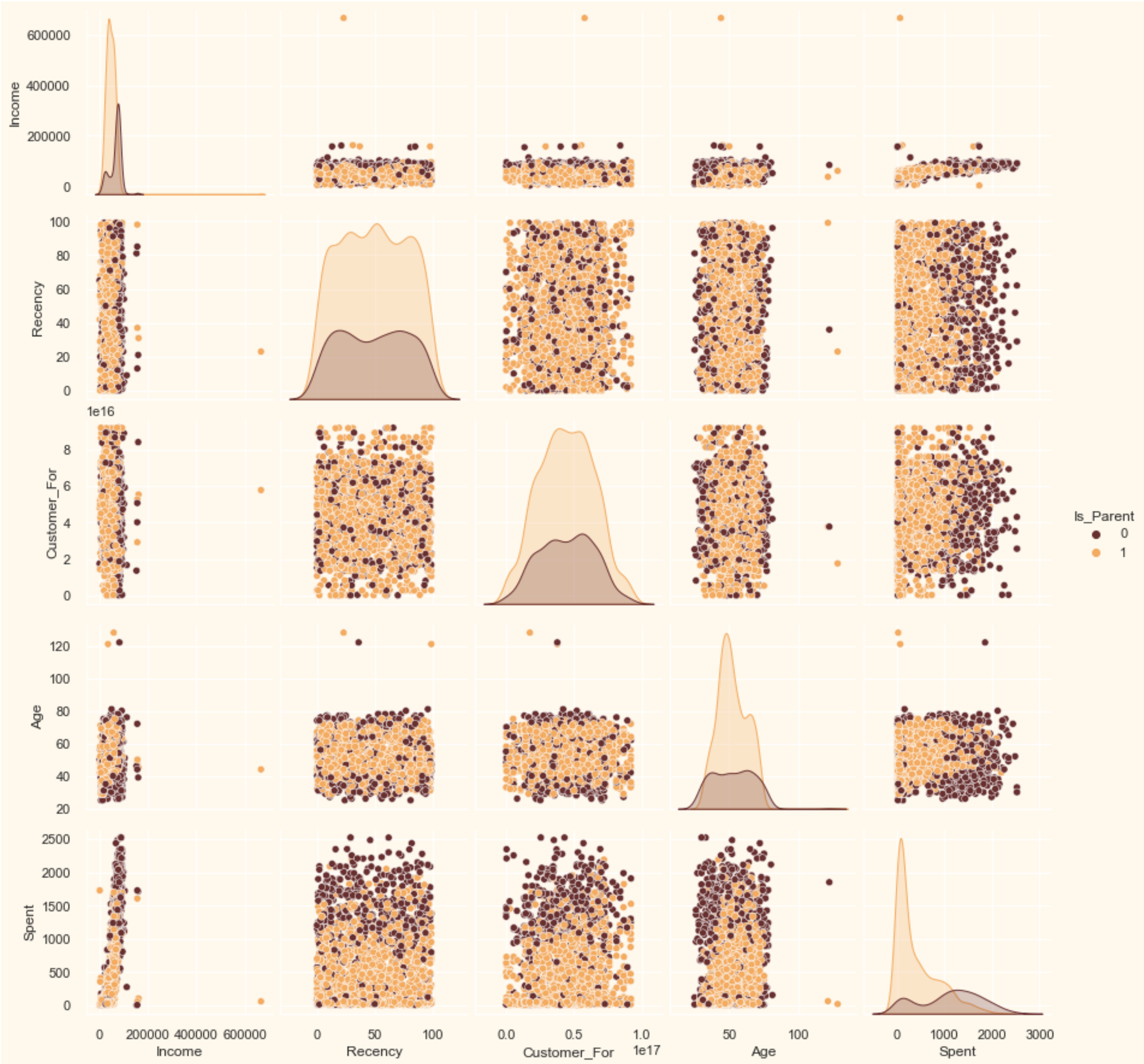
Out[70]:

	Income	Kidhome	Teenhome	Recency	Wines	Fruits	Meat
count	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
std	25173.076661	0.536896	0.544181	28.948352	337.327920	39.793917	224.283273
min	1730.000000	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
50%	51381.500000	0.000000	0.000000	49.000000	174.500000	8.000000	68.000000
75%	68522.000000	1.000000	1.000000	74.000000	505.000000	33.000000	232.250000
max	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000

8 rows × 28 columns

In [71]:
```
sns.set(rc={"axes.facecolor":"#FFF9ED","figure.facecolor":"#FFF9ED"})
pallet = ["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#F3AB60"]
cmap = colors.ListedColormap(["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78"
#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 576x396 with 0 Axes>
```

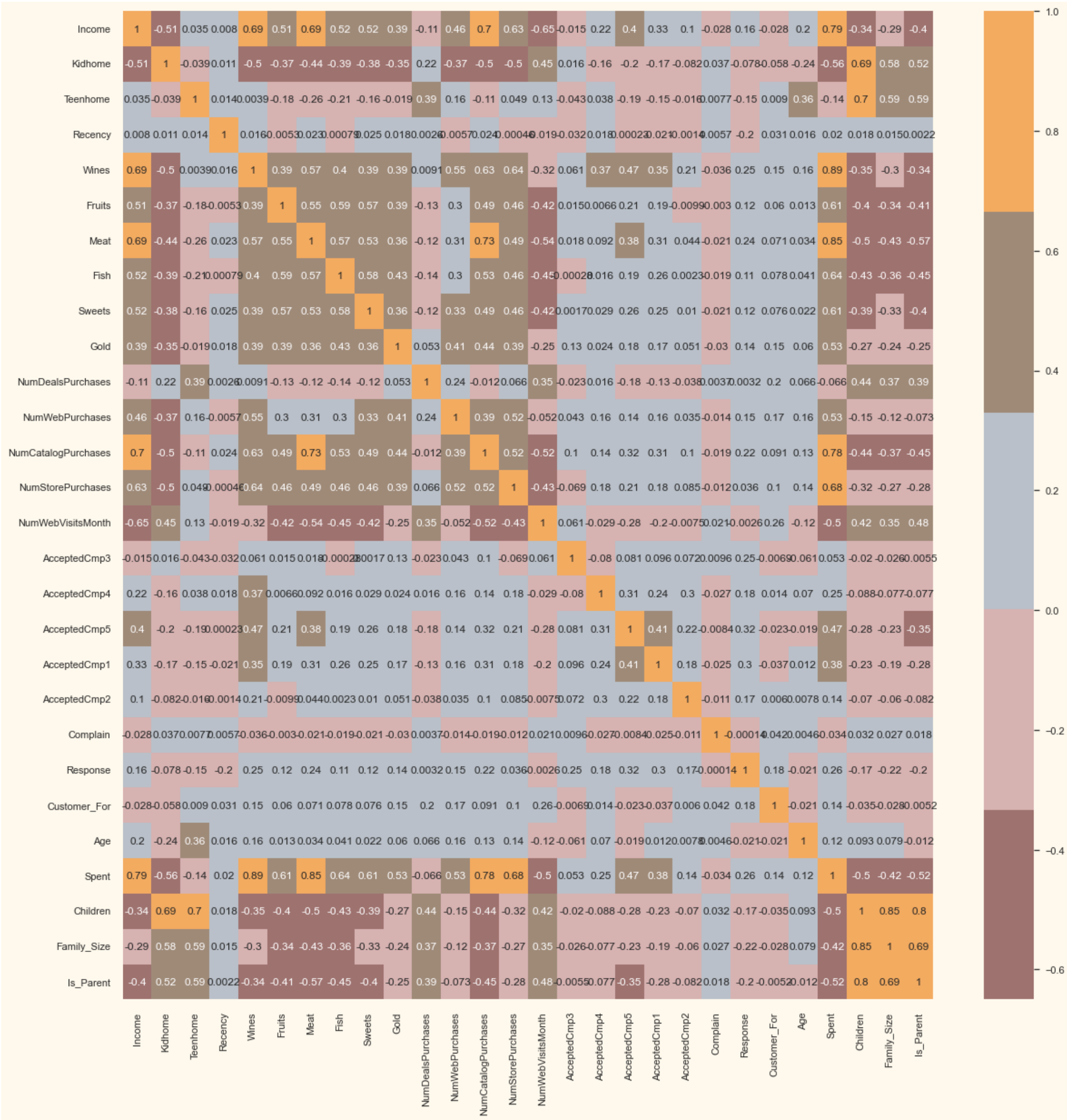


In [72]:
```
data = data[(data["Age"]<90)]
data = data[(data["Income"]<600000)]
print("The total number of data-points after removing the outliers are:", len(data)
```

```
The total number of data-points after removing the outliers are: 2212
```

In [73]:
```
cormat= data.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corrmat,annot=True, cmap=cmap, center=0)
```

Out[73]: <AxesSubplot:>



4) Data Preprocessing

1) Label encoding the categorical features

2) Scaling the features using the standard scaler

In [74]:
```
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 [75]:
```
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 [76]:
```
ds = data.copy()
# creating a subset of dataframe by dropping the features on deals accepted and pro
cols_del = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1','Accepte
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 [77]:
```
print("Dataframe to be used for further modelling:")
scaled_ds.head()
```

Dataframe to be used for further modelling:

Out[77]:

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

5 rows × 23 columns

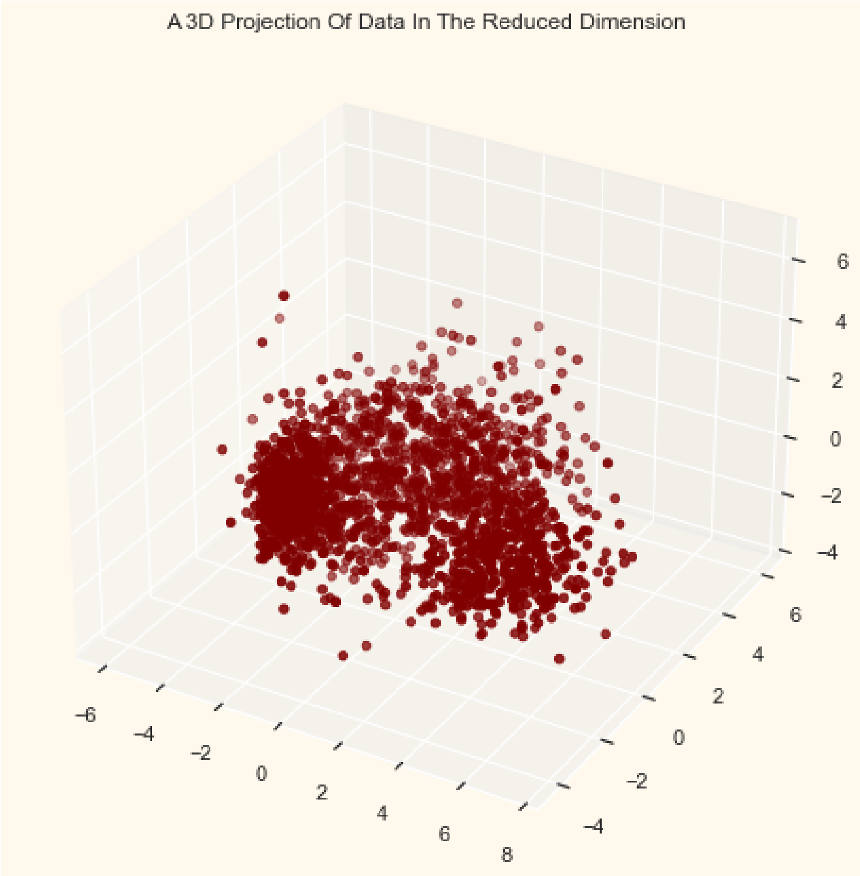
5) Dimensionality Reduction

In [78]:
```
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[78]:

	count	mean	std	min	25%	50%	75%	max
col1	2212.0	-1.033933e-16	2.878377	-5.969394	-2.538494	-0.780421	2.383290	7.444305
col2	2212.0	9.325472e-17	1.706839	-4.312196	-1.328316	-0.158123	1.242289	6.142721
col3	2212.0	4.692850e-17	1.221956	-3.530416	-0.829067	-0.022692	0.799895	6.611222

In [79]:
```
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()
```



6) Clustering

1) Elbow method for k value

2) K-Means Clustering

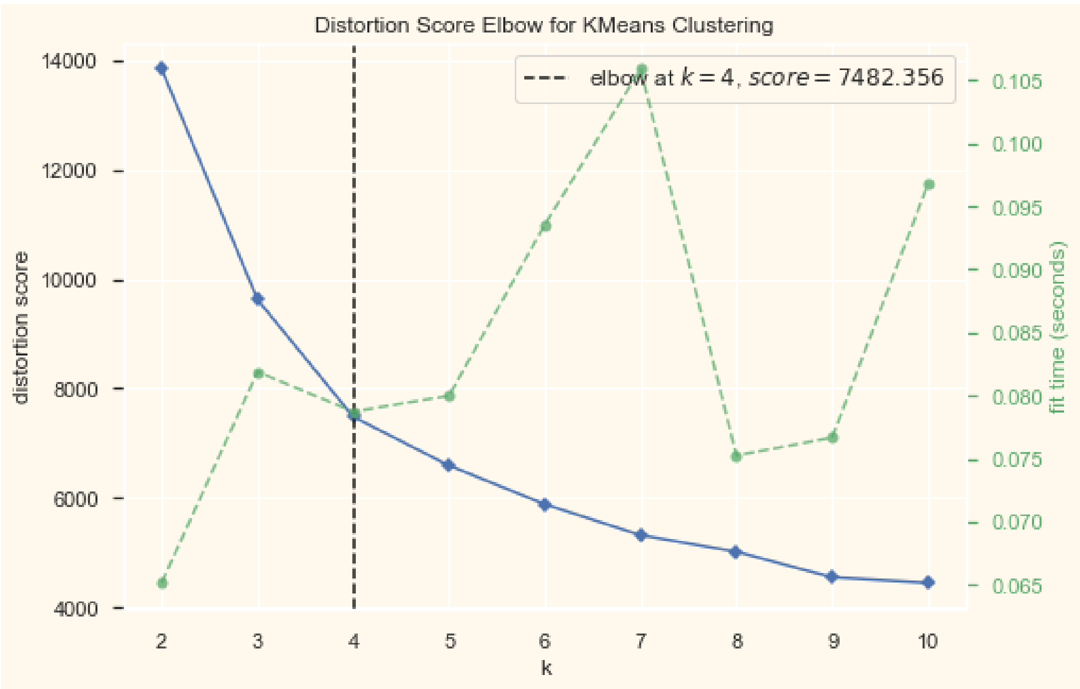
3) Agglomerative Clustering

4) Spectral Clustering

5) Ensemble Clustering using majority voting

In [80]:
```
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:
```



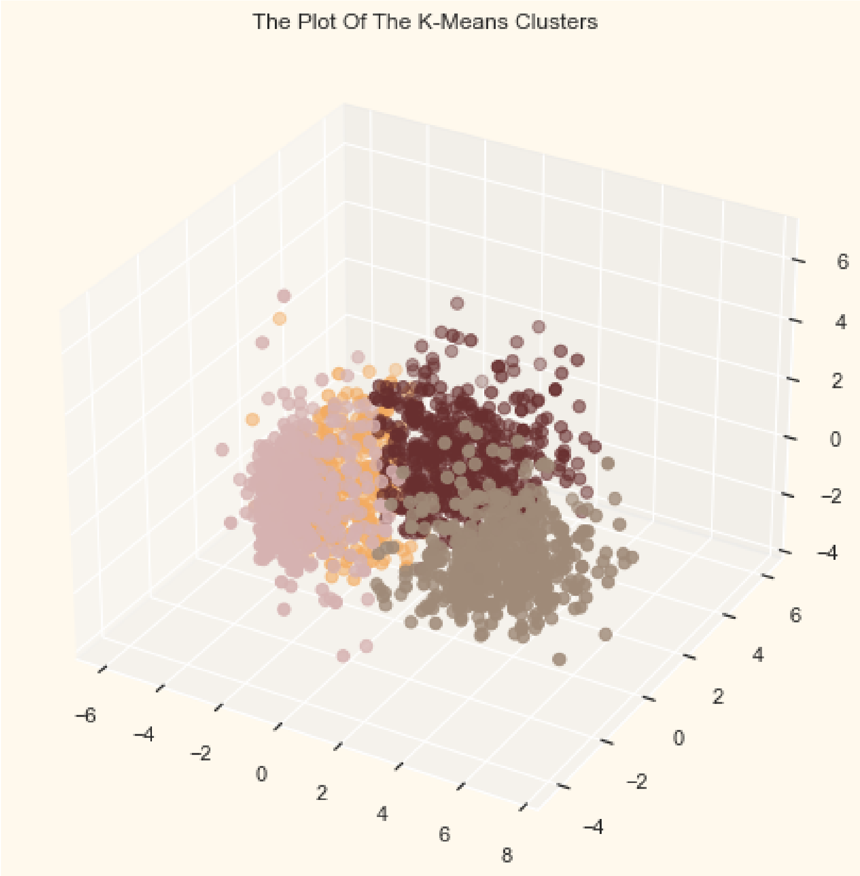
Out[80]:
```
<AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>
```

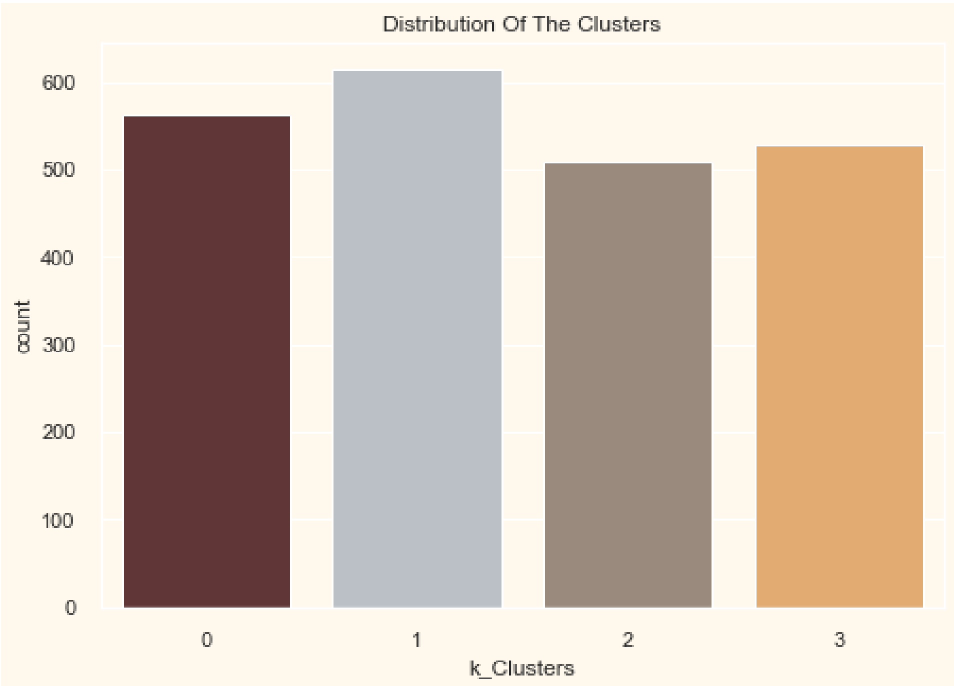
K-Means Clustering

In [81]:
```
k = KMeans(n_clusters=4)
# fit model and predict clusters
k1 = k.fit_predict(PCA_ds)
PCA_ds["k_Clusters"] = k1
#Adding the Clusters feature to the orignal dataframe.
data["k_Clusters"]= k1

print("K-Means Cluster Counts:")
print(PCA_ds['k_Clusters'].value_counts())
print()
```

```
K-Means Cluster Counts:
1    614
0    563
3    527
2    508
Name: k_Clusters, dtype: int64
```

In [82]:
```
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds["k_Clusters"], marker='o', cmap = cmap )
ax.set_title("The Plot Of The K-Means Clusters")
plt.show()

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



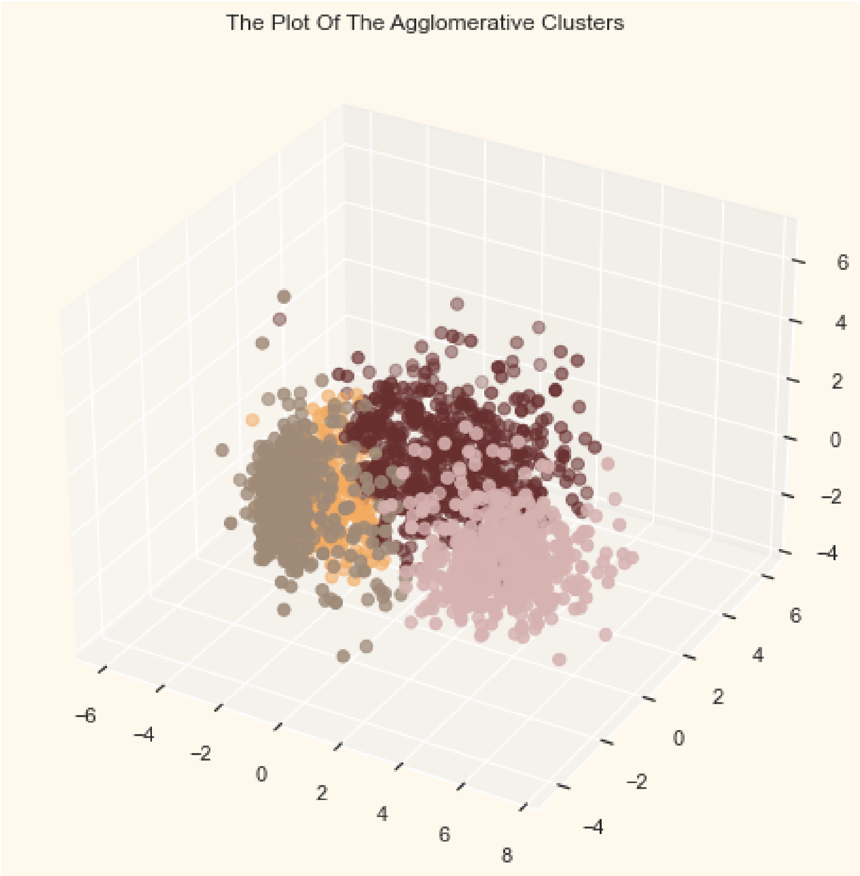


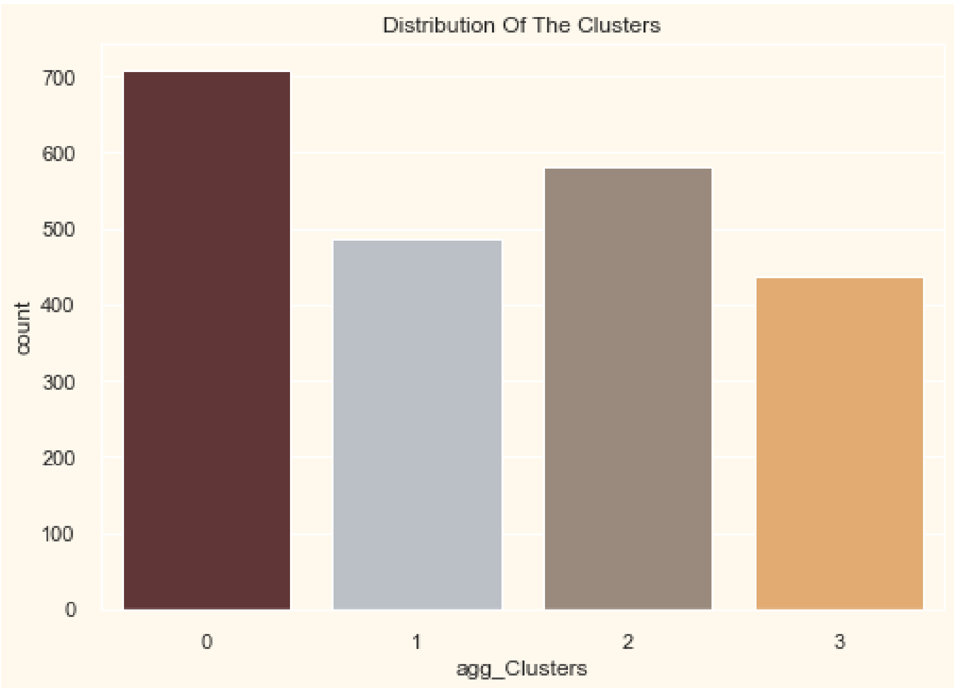
Agglomerative Clustering

In [83]:
```
AC = AgglomerativeClustering(n_clusters=4)
# fit model and predict clusters
yhat_AC = AC.fit_predict(PCA_ds.iloc[: ,:-1])
PCA_ds["agg_Clusters"] = yhat_AC
#Adding the Clusters feature to the orignal dataframe.
data["agg_Clusters"]= yhat_AC

print("Agglomerative Cluster Counts:")
print(PCA_ds['agg_Clusters'].value_counts())
print()
```

```
Agglomerative Cluster Counts:
0    708
2    580
1    487
3    437
Name: agg_Clusters, dtype: int64
```

In [84]:
```
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds["agg_Clusters"], marker='o', cmap = cmap )
ax.set_title("The Plot Of The Agglomerative Clusters")
plt.show()

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



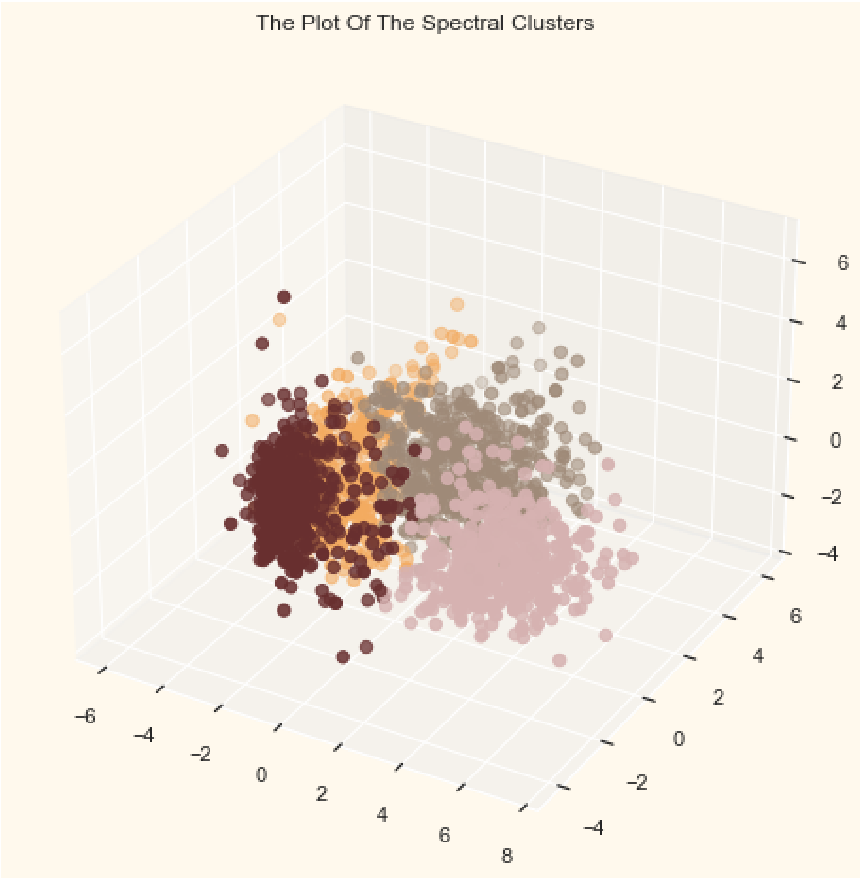


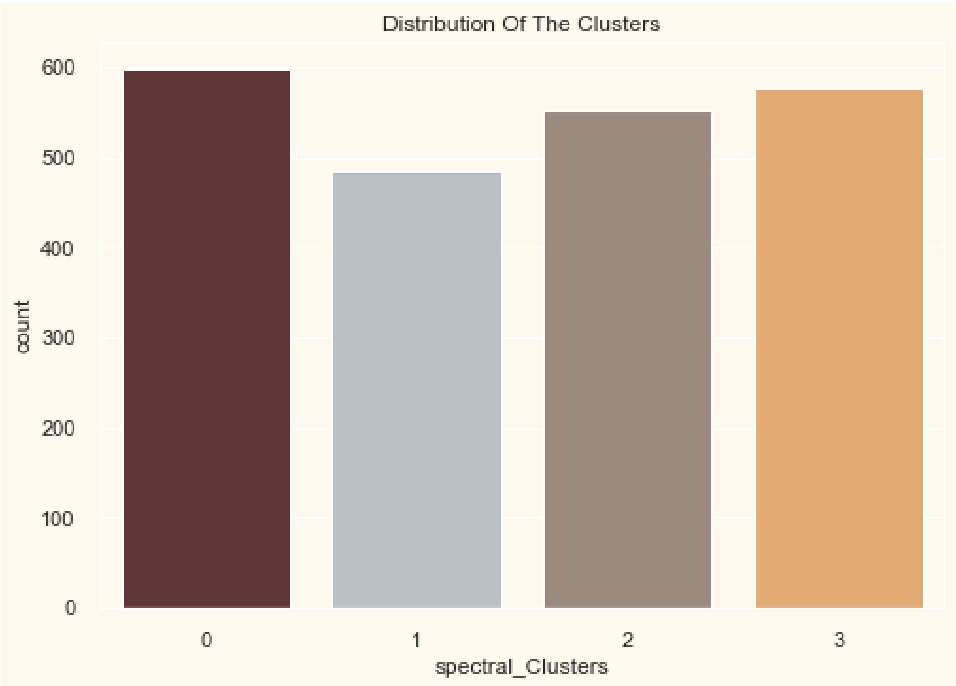
Spectral Clustering

In [85]:
```
SC = SpectralClustering(n_clusters=4)
# fit model and predict clusters
sc1 = SC.fit_predict(PCA_ds.iloc[: ,:-2])
PCA_ds["spectral_Clusters"] = sc1
#Adding the Clusters feature to the orignal dataframe.
data["spectral_Clusters"]= sc1

print("Spectral Cluster Counts:")
print(PCA_ds['spectral_Clusters'].value_counts())
print()
```

```
Spectral Cluster Counts:
0    598
3    578
2    552
1    484
Name: spectral_Clusters, dtype: int64
```

In [86]:
```
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds["spectral_Clusters"], marker='o', cmap = cmap )
ax.set_title("The Plot Of The Spectral Clusters")
plt.show()

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



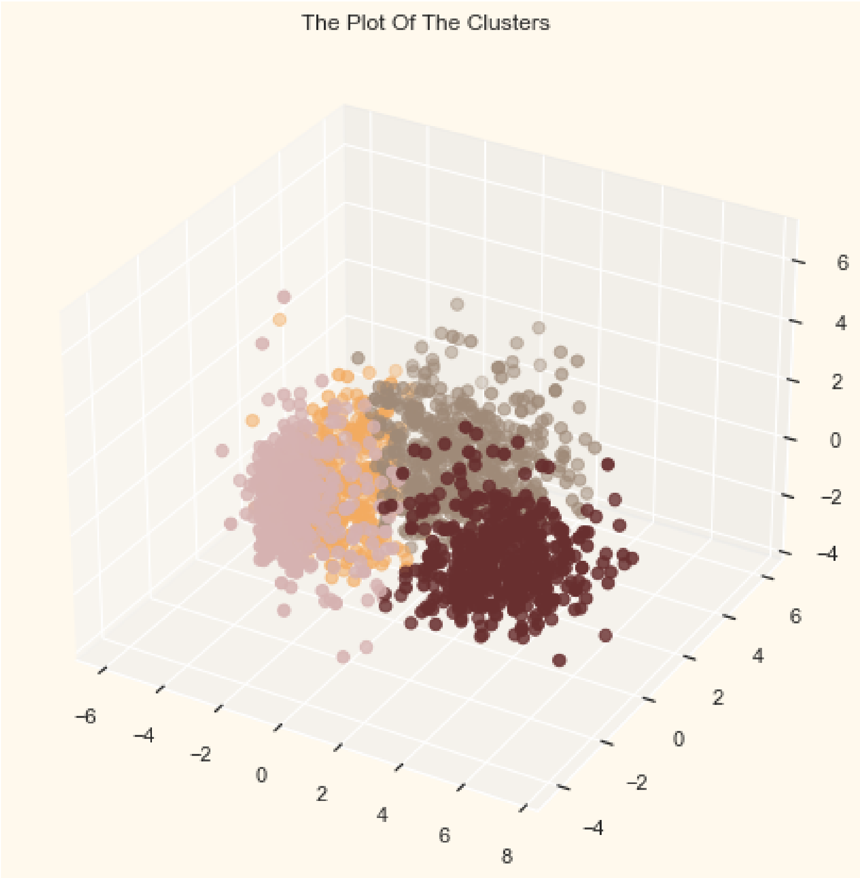


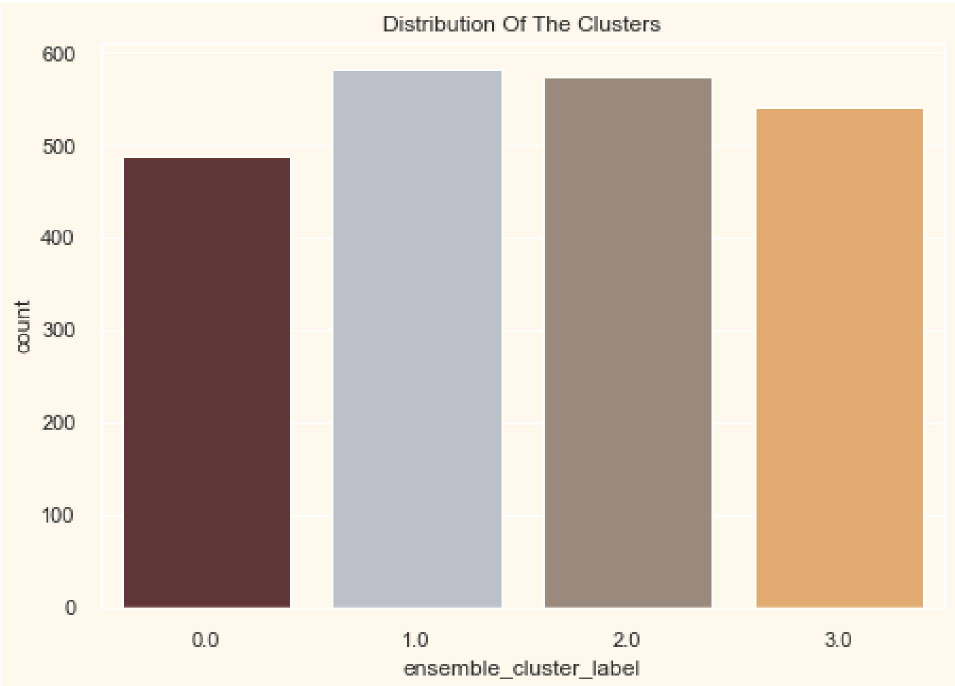
Ensemble Clustering

In [88]:
```
kmeans_labels = PCA_ds['k_Clusters']
agg_labels = PCA_ds['agg_Clusters']
spectral_labels = PCA_ds['spectral_Clusters']
cluster_labels_array = np.array([kmeans_labels, agg_labels, spectral_labels])
# Step 1: Calculate distance matrices for each base clustering algorithm
distance_matrices = []
for labels in cluster_labels_array:
    distance_matrix = squareform(pdist(labels.reshape(-1, 1), metric='hamming'))  #
    distance_matrices.append(distance_matrix)

# Step 2: Combine distance matrices
combined_distance_matrix = np.mean(distance_matrices, axis=0)

# Step 3: Convert combined distance matrix to condensed form
condensed_distance_matrix = squareform(combined_distance_matrix)

# Step 4: Perform hierarchical clustering
linkage_matrix = linkage(condensed_distance_matrix, method='average')  # or 'comple

# Step 5: Cut the dendrogram to obtain clusters
ensemble_cluster_labels = fcluster(linkage_matrix, t=4, criterion='maxclust')  # Ad
ensemble_cluster_labels_adjusted = ensemble_cluster_labels - 1

# Assign the ensemble labels to your DataFrame
PCA_ds['ensemble_cluster_label'] = ensemble_cluster_labels_adjusted
data["ensemble_cluster_label"] = PCA_ds['ensemble_cluster_label']

print("Ensemble Cluster Counts:")
print(PCA_ds['ensemble_cluster_label'].value_counts())
print()
```

```
Ensemble Cluster Counts:
1    593
2    579
3    548
0    492
Name: ensemble_cluster_label, dtype: int64
```

In [89]:
```
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds["ensemble_cluster_label"], marker='o', cmap = cm
ax.set_title("The Plot Of The Clusters")
plt.show()

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





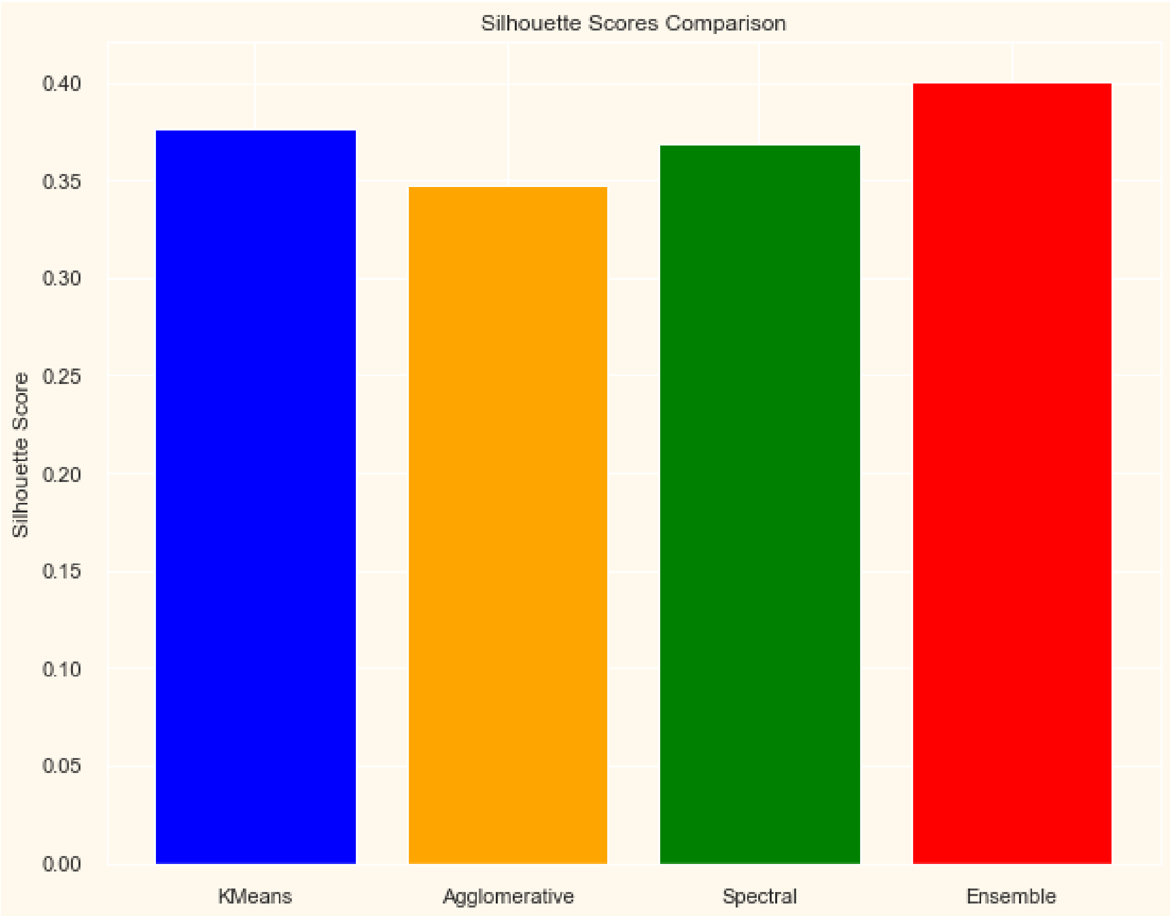
7) Model Evaluation

Silhouette Score

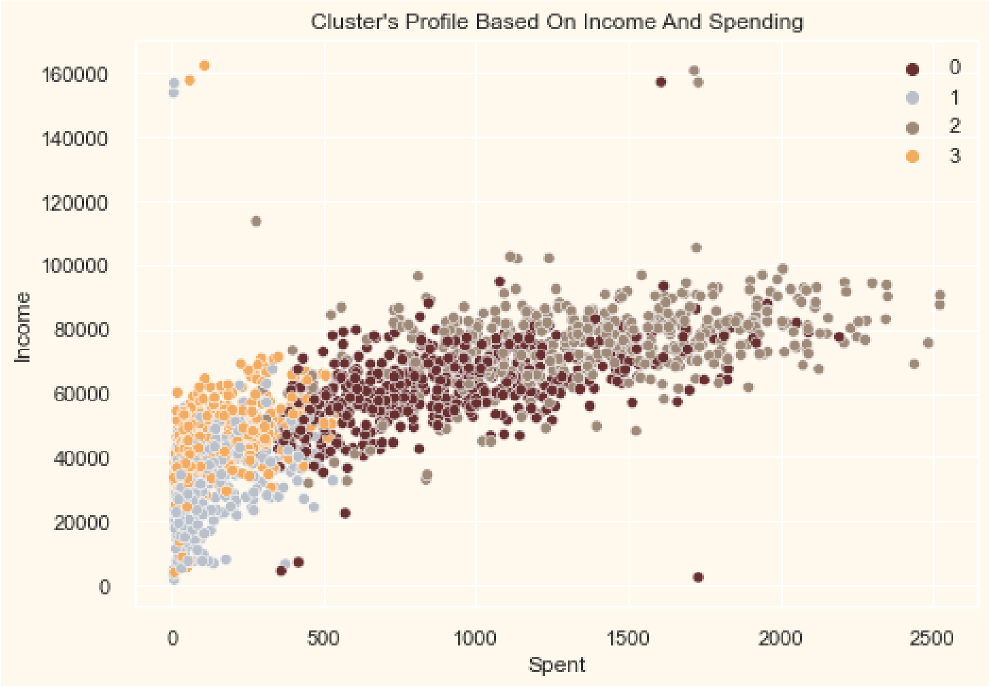
In [90]:
```
silhouette_kmeans = silhouette_score(PCA_ds.iloc[:, :-4], PCA_ds['k_Clusters'])
silhouette_agg = silhouette_score(PCA_ds.iloc[:, :-4], PCA_ds['agg_Clusters'])
silhouette_spectral = silhouette_score(PCA_ds.iloc[:, :-4], PCA_ds['spectral_Cluste
# Compute silhouette score for ensemble clustering
silhouette_scores_ensemble = silhouette_samples(PCA_ds.iloc[:, :-4], PCA_ds['ensemb
bottom_indices = np.argsort(silhouette_scores_ensemble)[:100]
PCA_ds_modified = PCA_ds.drop(index=bottom_indices)
silhouette_ensemble = silhouette_score(PCA_ds_modified.iloc[:, :-4], PCA_ds_modifie

# Print silhouette scores
print(f"Silhouette Score - KMeans: {silhouette_kmeans}")
print(f"Silhouette Score - Agglomerative: {silhouette_agg}")
print(f"Silhouette Score - Spectral: {silhouette_spectral}")
print(f"Silhouette Score - Ensemble: {silhouette_ensemble}")
```

```
Silhouette Score - KMeans: 0.3759726431626535
Silhouette Score - Agglomerative: 0.3472956525154227
Silhouette Score - Spectral: 0.3687240653254949
Silhouette Score - Ensemble: 0.4008917470215087
```

In [91]:
```
algorithms = ['KMeans', 'Agglomerative', 'Spectral', 'Ensemble']
silhouette_scores = [silhouette_kmeans, silhouette_agg, silhouette_spectral, silhou
plt.figure(figsize=(10, 8))
plt.bar(algorithms, silhouette_scores, color=['blue', 'orange', 'green', 'red'])
plt.title('Silhouette Scores Comparison')
plt.ylabel('Silhouette Score')
plt.show()
```



In [92]:
```
pl = sns.scatterplot(data = data,x=data["Spent"], y=data["Income"],hue=data["k_Clus
pl.set_title("Cluster's Profile Based On Income And Spending")
plt.legend()
plt.show()
```



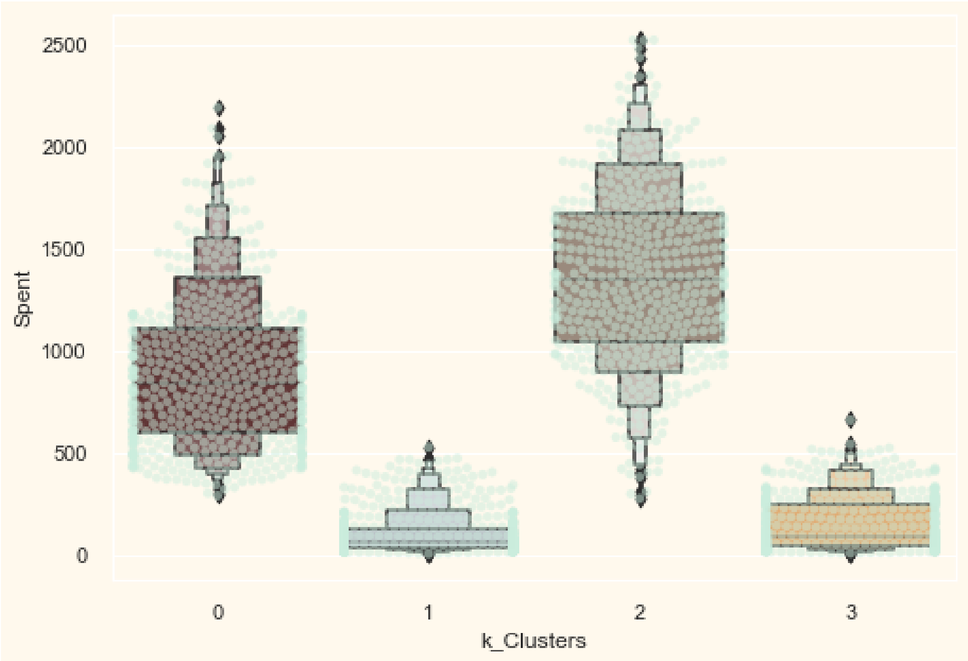
group 0: high spending & high income

group 1: low spending & low income

group 2: high spending & average income

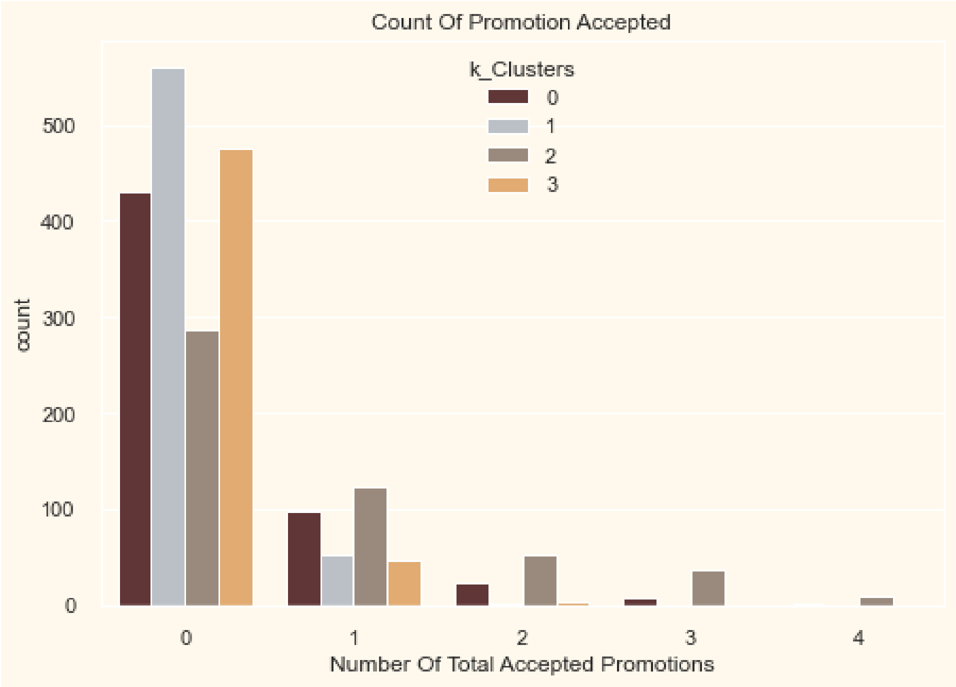
group 3: high spending & low income

In [93]:
```
plt.figure()
pl=sns.swarmplot(x=data["k_Clusters"], y=data["Spent"], color= "#CBEDDD", alpha=0.5
pl=sns.boxenplot(x=data["k_Clusters"], y=data["Spent"], palette=pal)
plt.show()
```



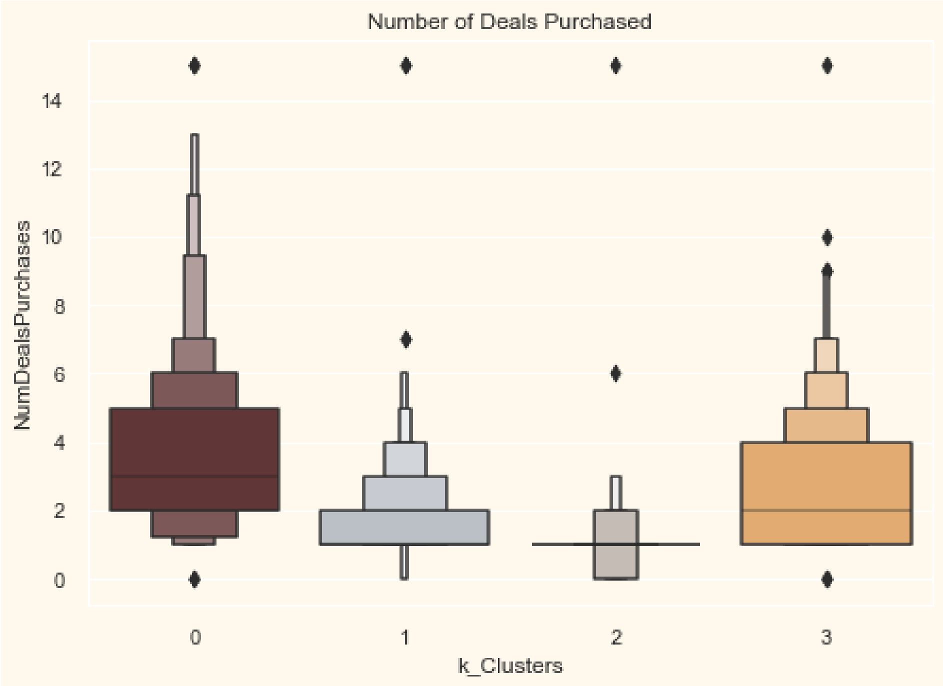
From the above plot, it can be clearly seen that cluster 0 is our biggest set of customers closely followed by cluster 2.

In [94]:
```
data["Total_Promos"] = data["AcceptedCmp1"]+ data["AcceptedCmp2"]+ data["AcceptedCm
#Plotting count of total campaign accepted.
plt.figure()
pl = sns.countplot(x=data["Total_Promos"],hue=data["k_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 [95]:
```
plt.figure()
pl=sns.boxenplot(y=data["NumDealsPurchases"],x=data["k_Clusters"], palette= pal)
pl.set_title("Number of Deals Purchased")
plt.show()
```

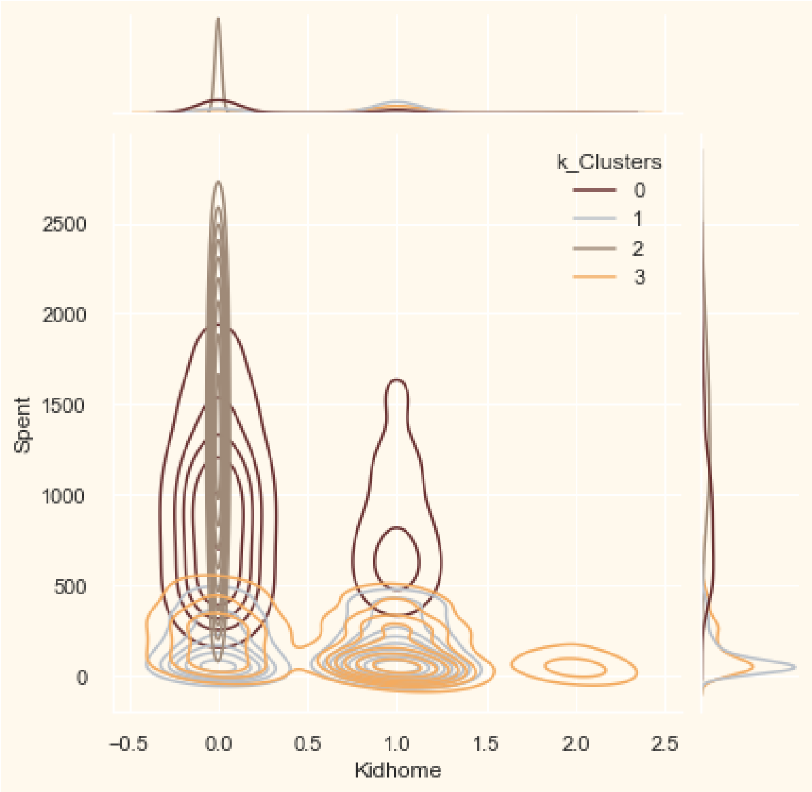


It has best outcome with cluster 2 and cluster 3. However, our star customers cluster 0 are not much into the deals.

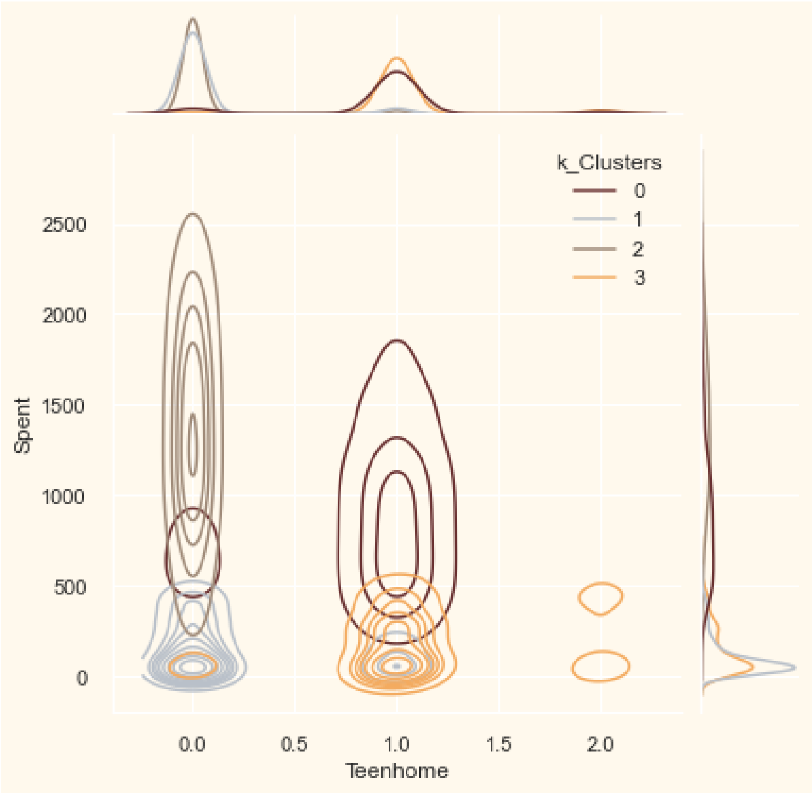
8) Profiling

In [96]:
```
Personal = [ "Kidhome","Teenhome","Customer_For"]

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

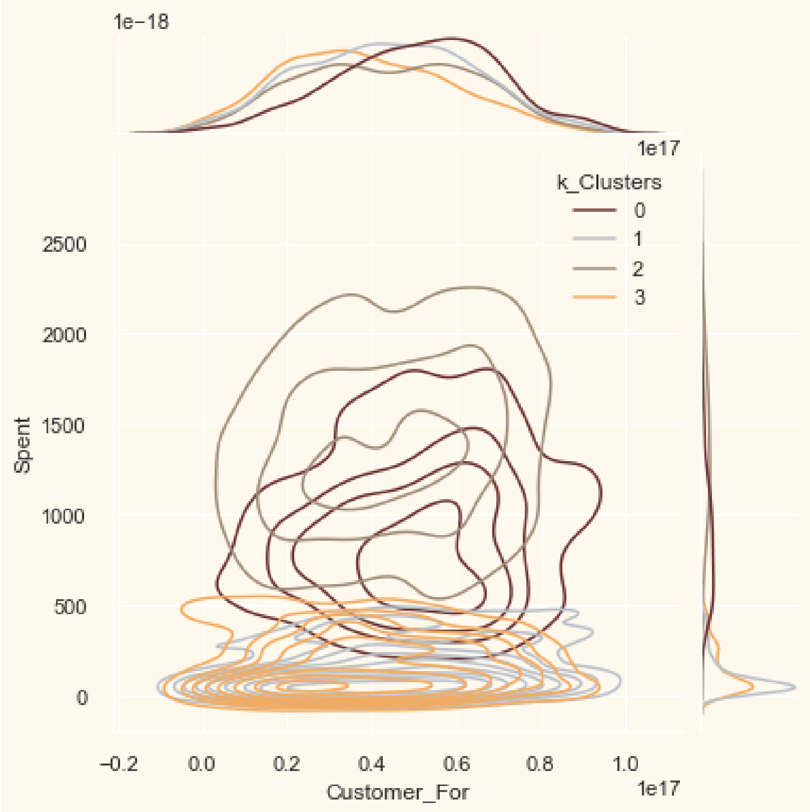
```
<Figure size 576x396 with 0 Axes>
```




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

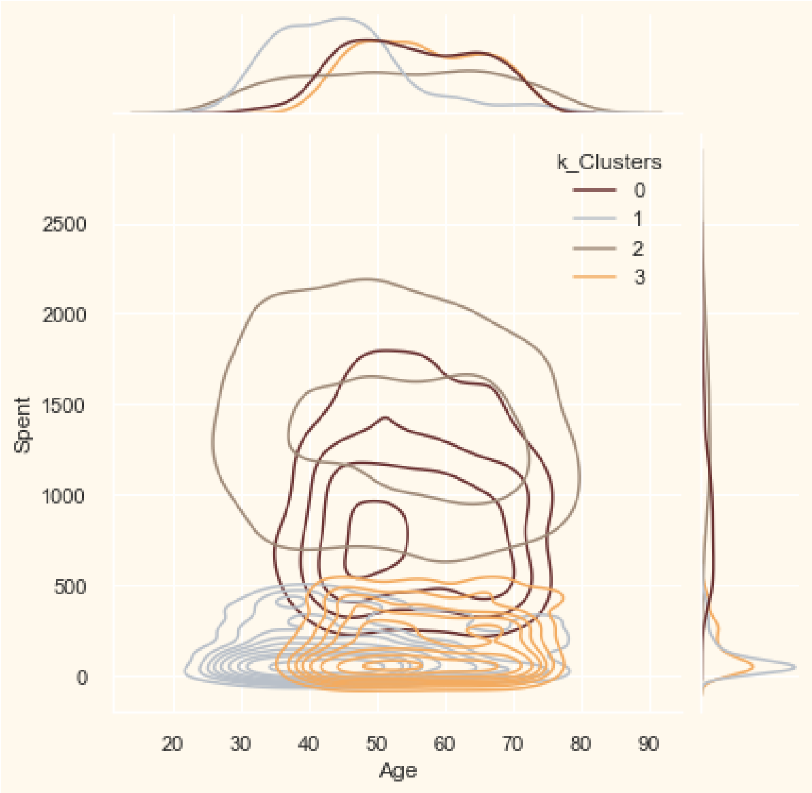



```
<Figure size 576x396 with 0 Axes>
```

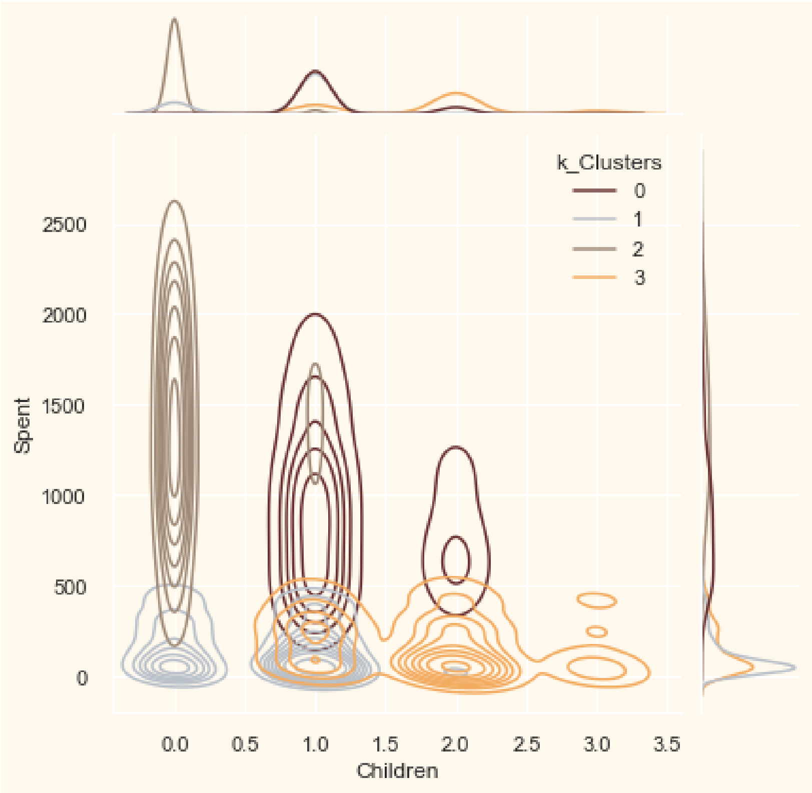

1e−18
1e17
k_Clusters
0
1
2
3
2500
2000
1500
1000
500
0
Spent
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
Customer_For
1e17

In [97]:
```
Personal = [ "Age", "Children", "Family_Size" ]

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

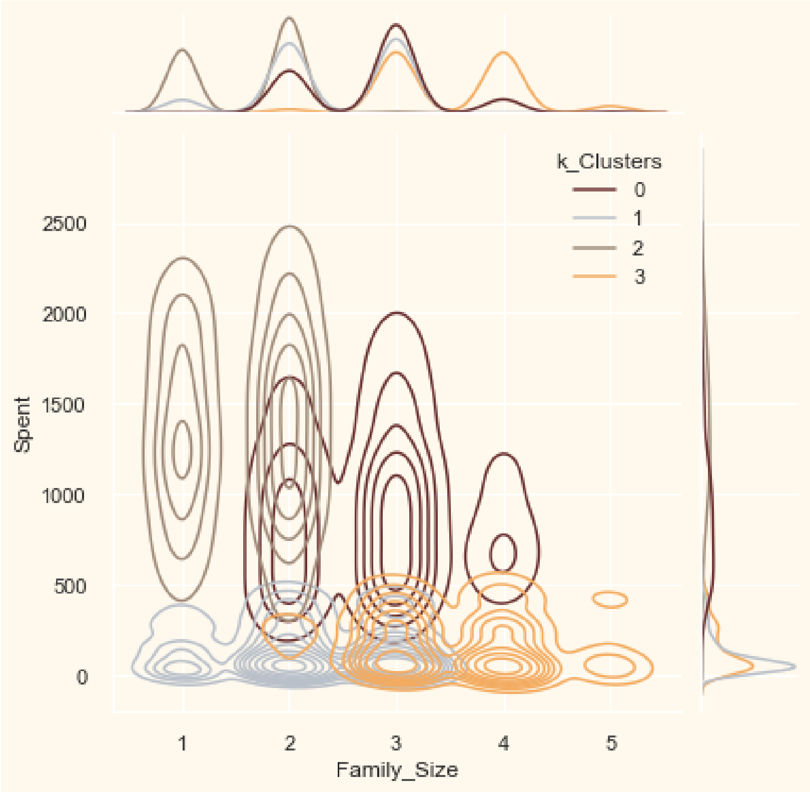
```
<Figure size 576x396 with 0 Axes>
```




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

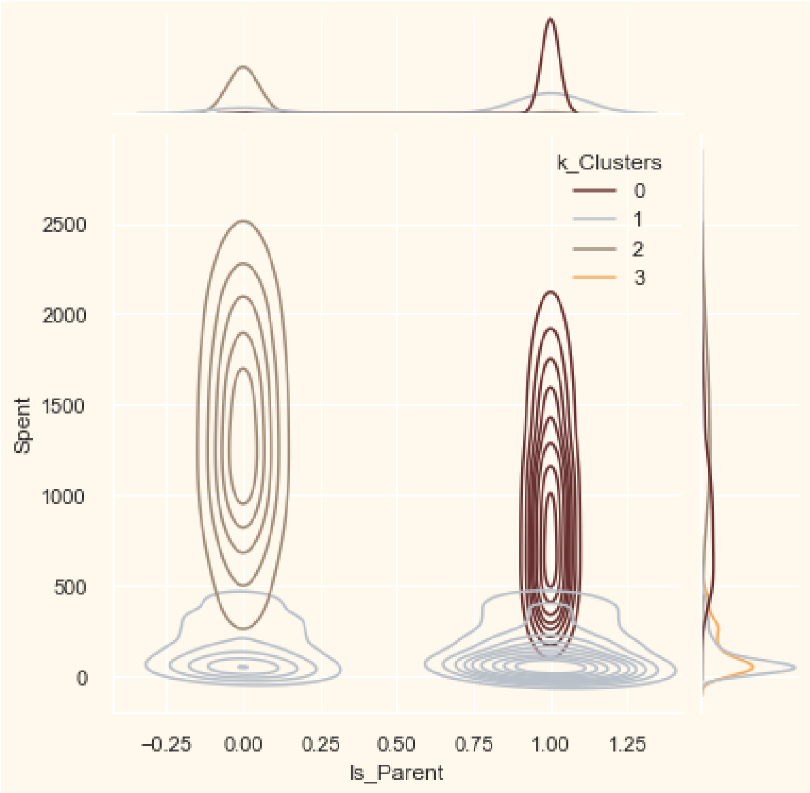



```
<Figure size 576x396 with 0 Axes>
```

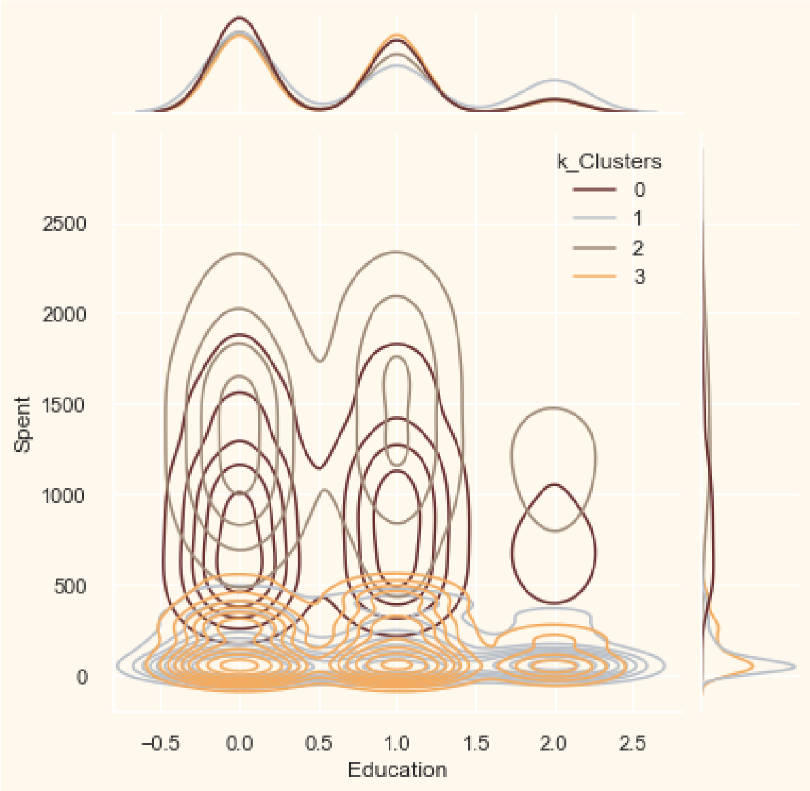

k_Clusters
0
1
2
3
Spent
2500
2000
1500
1000
500
0
1
2
3
4
5
Family_Size

In [98]:
```
if not sys.warnoptions:
    warnings.simplefilter("ignore")
np.random.seed(42)

Personal = [ "Is_Parent", "Education","Living_With" ]

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

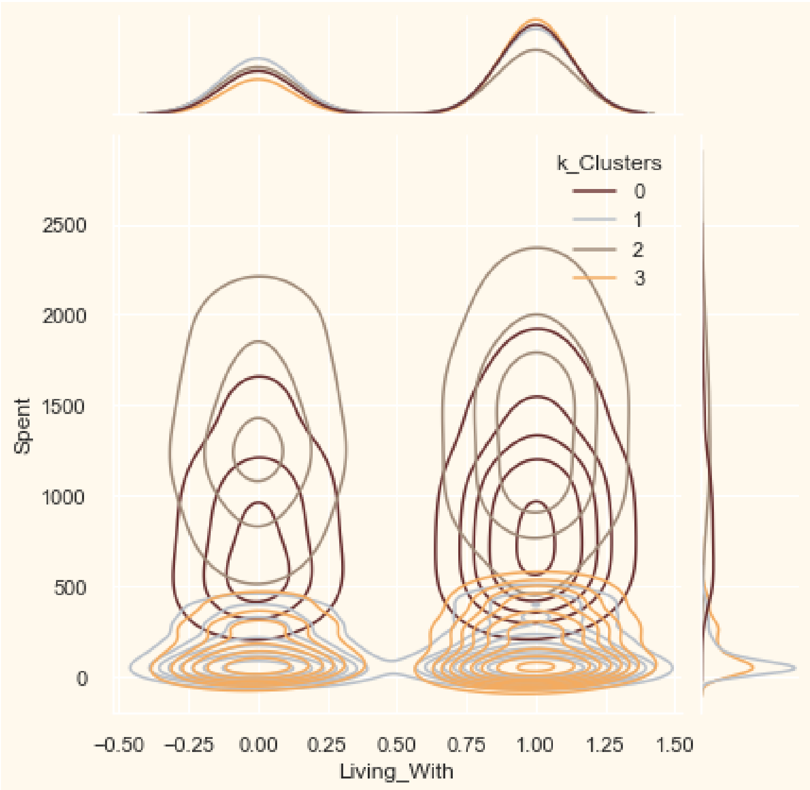
```
<Figure size 576x396 with 0 Axes>
```




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




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



Cluster 0

Definetly not a parent

max are only 2 members in the family

majority of couples over single persons

Span all ages

A high income group

Cluster 1

majority of people are parents

max 3 members in the family

majority have one kid

Relatively younger

Cluster 2

Definetly a parent

Max have 4 members in family and atleast 2

most have a teenagers at home

Relatively older

Cluster 3

Definetly a parent

At max 5 members in family and at least 2

Majority of them have teenagers at home

Relatively older

Low-income group

In [ ]: