In [82]:
```
# Prasad Jadhav
```

Customer Personality Analysis Project

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

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler

from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans, AgglomerativeClustering

from sklearn.metrics import confusion_matrix

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn import metrics

from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
from sklearn.cluster import SpectralClustering
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_samples
from sklearn.metrics import silhouette_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

import pickle
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
dataset = pd.read_csv('marketing_campaign.csv',sep='\t')
pd.set_option('display.max_columns',100)
dataset['ID'].drop
dataset.shape
```

Out[3]:
```
(2240, 29)
```

In [5]:
```
dataset.head()
```

Out[5]:

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

In [5]:
```
dataset.tail()
```

Out[5]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer
2235	10870	1967	Graduation	Married	61223.0	0	1	13-06-2013
2236	4001	1946	PhD	Together	64014.0	2	1	10-06-2014
2237	7270	1981	Graduation	Divorced	56981.0	0	0	25-01-2014
2238	8235	1956	Master	Together	69245.0	0	1	24-01-2014
2239	9405	1954	PhD	Married	52869.0	1	1	15-10-2012

In [6]:
```
print('Number of Rows:',dataset.shape[0])
print('Number of Columns:',dataset.shape[1])
```

```
Number of Rows: 2240
Number of Columns: 29
```

In [7]:
```
dataset.dtypes
```

Out[7]:
```
ID                       int64
Year_Birth               int64
Education               object
Marital_Status          object
Income                 float64
Kidhome                  int64
Teenhome                 int64
Dt_Customer             object
Recency                  int64
MntWines                 int64
MntFruits                int64
MntMeatProducts          int64
MntFishProducts          int64
MntSweetProducts         int64
MntGoldProds             int64
NumDealsPurchases        int64
NumWebPurchases          int64
NumCatalogPurchases      int64
NumStorePurchases        int64
NumWebVisitsMonth        int64
AcceptedCmp3             int64
AcceptedCmp4             int64
AcceptedCmp5             int64
AcceptedCmp1             int64
AcceptedCmp2             int64
Complain                 int64
Z_CostContact            int64
Z_Revenue                int64
Response                 int64
dtype: object
```

In [8]:
```
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 [9]:
```
dataset.isnull().sum()
```

Out[9]:
```
ID                     0
Year_Birth             0
Education              0
Marital_Status         0
Income                24
Kidhome                0
Teenhome               0
Dt_Customer            0
Recency                0
MntWines               0
MntFruits              0
MntMeatProducts        0
MntFishProducts        0
MntSweetProducts       0
MntGoldProds           0
NumDealsPurchases      0
NumWebPurchases        0
NumCatalogPurchases    0
NumStorePurchases      0
NumWebVisitsMonth      0
AcceptedCmp3           0
AcceptedCmp4           0
AcceptedCmp5           0
AcceptedCmp1           0
AcceptedCmp2           0
Complain               0
Z_CostContact          0
Z_Revenue              0
Response               0
dtype: int64
```

In [10]:
```
dataset.duplicated().sum()
```

Out[10]:
```
0
```

In [11]:
```
dataset = dataset.drop_duplicates()
dataset = dataset.dropna()
```

In [12]:
```
dataset.cov()
```

Out[12]:

	ID	Year_Birth	Income	Kidhome	Teenhome	
ID	1.055845e+07	91.725792	1.071169e+06	3.028930	-5.357445	-41
Year_Birth	9.172579e+01	143.653507	-4.881462e+04	1.503310	-2.287966	
Income	1.071169e+06	-48814.621201	6.336838e+08	-5793.603194	262.102701	-28
Kidhome	3.028930e+00	1.503310	-5.793603e+03	0.288258	-0.011649	
Teenhome	-5.357445e+00	-2.287966	2.621027e+02	-0.011649	0.296133	
Recency	-4.174185e+03	-5.653712	-2.892837e+03	0.178614	0.217990	8
MntWines	-2.310976e+04	-644.670673	4.913652e+06	-90.072543	0.687766	1
MntFruits	9.473270e+02	-8.464241	4.315893e+05	-7.977684	-3.823374	
MntMeatProducts	-4.301363e+03	-90.582220	3.300781e+06	-52.894367	-31.870181	1
MntFishProducts	-4.268427e+03	-26.528317	6.048869e+05	-11.431713	-6.115193	
MntSweetProducts	-7.922359e+02	-9.946052	4.556893e+05	-8.336016	-3.644400	
MntGoldProds	-1.880985e+03	-39.875318	4.251102e+05	-9.876741	-0.560760	
NumDealsPurchases	-2.538575e+02	-1.352697	-4.024231e+03	0.224036	0.404342	
NumWebPurchases	-1.645566e+02	-5.028016	2.676286e+04	-0.547403	0.241750	
NumCatalogPurchases	-2.162818e+01	-4.271300	4.340648e+04	-0.792748	-0.179482	
NumStorePurchases	-1.380552e+02	-4.982937	4.331890e+04	-0.875021	0.087986	
NumWebVisitsMonth	-6.142249e+01	3.601796	-3.376809e+04	0.582689	0.173215	
AcceptedCmp3	-3.044993e+01	0.193352	-1.063120e+02	0.002252	-0.006042	
AcceptedCmp4	-2.036296e+01	-0.207470	1.215450e+03	-0.022778	0.005468	
AcceptedCmp5	-5.976632e+00	0.033000	2.201852e+03	-0.028700	-0.027033	
AcceptedCmp1	-1.450076e+01	-0.028215	1.706909e+03	-0.022905	-0.019309	
AcceptedCmp2	-5.631422e+00	-0.009306	2.547302e+02	-0.005081	-0.000976	
Complain	1.075466e+01	-0.035318	-6.641277e+01	0.002132	0.000174	
Z_CostContact	0.000000e+00	0.000000	0.000000e+00	0.000000	0.000000	
Z_Revenue	0.000000e+00	0.000000	0.000000e+00	0.000000	0.000000	
Response	-2.495876e+01	0.101493	1.197060e+03	-0.014950	-0.029934	

In [13]:
```
dataset.corr()
```

Out[13]:

	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	MntWine
ID	1.000000	0.002355	0.013095	0.001736	-0.003030	-0.044376	-0.0210
Year_Birth	0.002355	1.000000	-0.161791	0.233615	-0.350791	-0.016295	-0.1594
Income	0.013095	-0.161791	1.000000	-0.428669	0.019133	-0.003970	0.5786
Kidhome	0.001736	0.233615	-0.428669	1.000000	-0.039869	0.011492	-0.4973
Teenhome	-0.003030	-0.350791	0.019133	-0.039869	1.000000	0.013838	0.0037
Recency	-0.044376	-0.016295	-0.003970	0.011492	0.013838	1.000000	0.0157
MntWines	-0.021084	-0.159451	0.578650	-0.497336	0.003747	0.015721	1.0000
MntFruits	0.007326	-0.017747	0.430842	-0.373396	-0.176558	-0.005844	0.3870
MntMeatProducts	-0.005902	-0.033697	0.584633	-0.439261	-0.261122	0.022518	0.5688
MntFishProducts	-0.023992	-0.040425	0.438871	-0.388884	-0.205242	0.000551	0.3977
MntSweetProducts	-0.005936	-0.020204	0.440744	-0.378026	-0.163056	0.025110	0.3903
MntGoldProds	-0.011172	-0.064208	0.325916	-0.355029	-0.019887	0.017663	0.3927
NumDealsPurchases	-0.040612	-0.058668	-0.083101	0.216913	0.386246	0.002115	0.0088
NumWebPurchases	-0.018476	-0.153051	0.387878	-0.371977	0.162077	-0.005641	0.5537
NumCatalogPurchases	-0.002274	-0.121764	0.589162	-0.504501	-0.112692	0.024081	0.6347
NumStorePurchases	-0.013070	-0.127891	0.529362	-0.501349	0.049737	-0.000434	0.6400
NumWebVisitsMonth	-0.007794	0.123904	-0.553088	0.447477	0.131240	-0.018564	-0.3219
AcceptedCmp3	-0.035890	0.061784	-0.016174	0.016066	-0.042522	-0.032257	0.0614
AcceptedCmp4	-0.023933	-0.066109	0.184400	-0.162026	0.038376	0.017566	0.3731
AcceptedCmp5	-0.007064	0.010575	0.335943	-0.205305	-0.190791	-0.000482	0.4735
AcceptedCmp1	-0.018219	-0.009611	0.276820	-0.174163	-0.144855	-0.021061	0.3514
AcceptedCmp2	-0.014994	-0.006717	0.087545	-0.081868	-0.015521	-0.001400	0.2061
Complain	0.034154	-0.030407	-0.027225	0.040978	0.003307	0.013637	-0.0394
Z_CostContact	NaN	NaN	NaN	NaN	NaN	NaN	Na
Z_Revenue	NaN	NaN	NaN	NaN	NaN	NaN	Na
Response	-0.021491	0.023692	0.133047	-0.077909	-0.153901	-0.199766	0.2462

In [14]:
```
dataset.describe()
```

Out[14]:

	ID	Year_Birth	Income	Kidhome	Teenhome	Recency	Mnt
count	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.(
mean	5588.353339	1968.820397	52247.251354	0.441787	0.505415	49.012635	305.(
std	3249.376275	11.985554	25173.076661	0.536896	0.544181	28.948352	337.3
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	0.000000	0.(
25%	2814.750000	1959.000000	35303.000000	0.000000	0.000000	24.000000	24.(
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000	49.000000	174.5
75%	8421.750000	1977.000000	68522.000000	1.000000	1.000000	74.000000	505.(
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	99.000000	1493.(

In [15]:
```
dataset['Dt_Customer'] = pd.to_datetime(dataset['Dt_Customer'])
dates = []
for i in dataset['Dt_Customer']:
    i = i.date()
    dates.append(i)
# Dates of the Newest and Oldest Recorded Customer
print('The Newest Customers Enrolment Date in the Records:', max(dates))
print('The Oldest Customers Enrolment Date in the Records:', min(dates))
```

```
The Newest Customers Enrolment Date in the Records: 2014-12-06
The Oldest Customers 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 [16]:
```
print('Total Categories in the Feature Marital Status:\n', dataset['Marital_S
print('Total Categories in the Feature Education:\n', dataset['Education'].va
```

```
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 the next bit, I will be performing the following steps to engineer some new features:

- 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 [17]:
```
dataset['Age'] = 2015 - dataset['Year_Birth']
```

In [18]:
```
# Create Another Feature Spent Indicating the Total Amount Spent by the Custom
dataset['Spent'] = dataset['MntWines'] + dataset['MntFruits'] + dataset['MntM
```

In [19]:
```
# Create Another Feature Living_With out of Marital_Status to Extract the Livi
dataset['Living_With'] = dataset['Marital_Status'].replace({'Married':'Partne
```

In [20]:
```
# Create Feature Children to Indicate Total Children in Household that is Kids
dataset['Children'] = dataset['Kidhome'] + dataset['Teenhome']
```

In [21]:
```
# To Get Further Clarity of Household Creating Feature Indicating Family_Size
dataset['Family_Size'] = dataset['Living_With'].replace({'Alone': 1, 'Partner
```

In [22]:
```
# Create Feature Is_Parent to Indicate Parenthood Status
dataset['Is_Parent'] = np.where(dataset.Children > 0, 1, 0)
```

In [6]:
```
# Segmenting Education Levels in Three Groups
dataset['Education'] = dataset['Education'].replace({'Basic':'Undergraduate',
```

In [8]:
```
dataset['Education'].value_counts()
```

Out[8]:
```
Graduate         1127
Postgraduate      856
Undergraduate     257
Name: Education, dtype: int64
```

In [24]:
```
# Dropping Some of the Redundant Features
to_drop = ['Marital_Status', 'Dt_Customer', 'Z_CostContact', 'Z_Revenue', 'Ye
dataset = dataset.drop(to_drop, axis=1)
```

In [25]:
```
dataset.describe(include=object)
```

Out[25]:

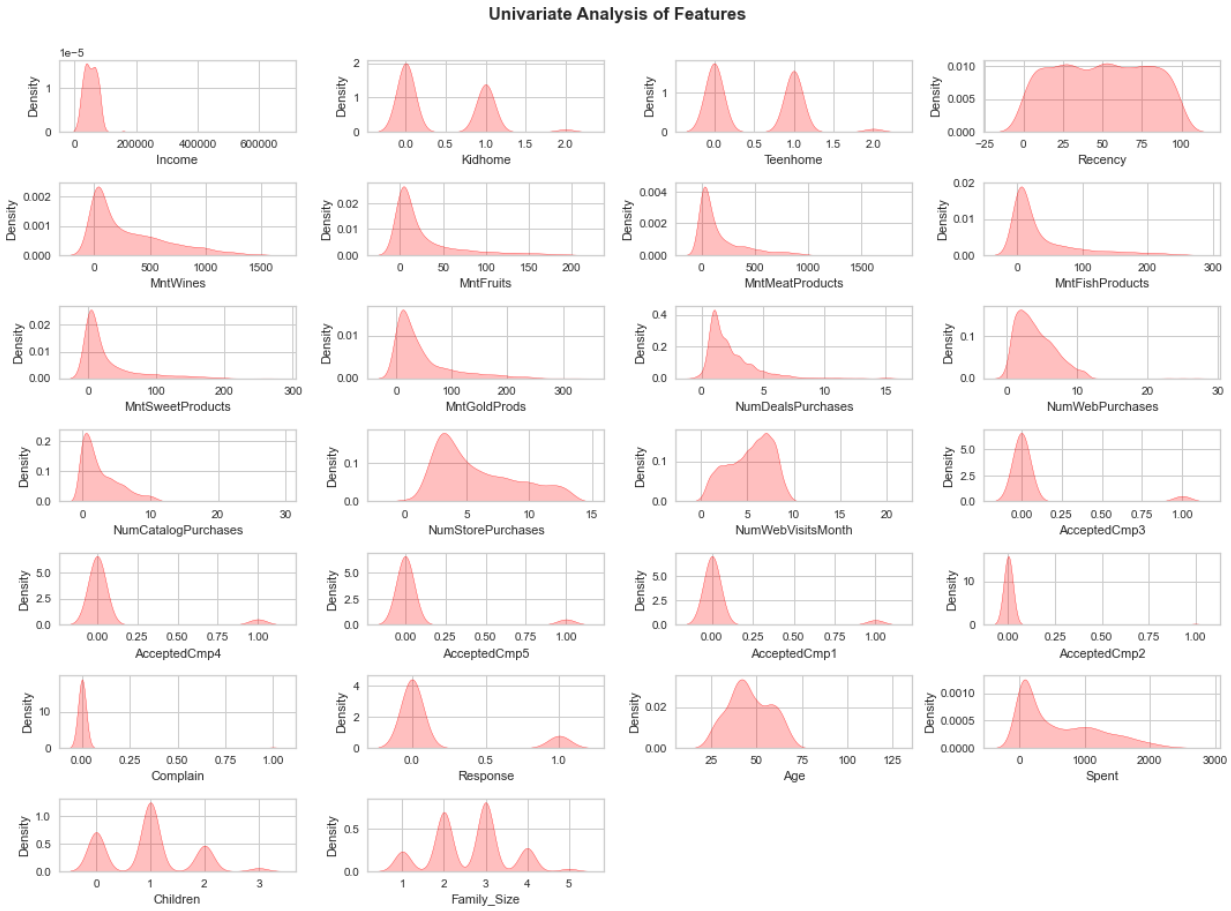
	Education	Living_With
count	2216	2216
unique	3	2
top	Graduate	Partner
freq	1116	1430

In [26]:
```
dataset['Income'] = dataset['Income'].astype('int64')
```

In [27]:
```
int_cols = [x for x in dataset.columns if dataset[x].dtypes=='int64']
```

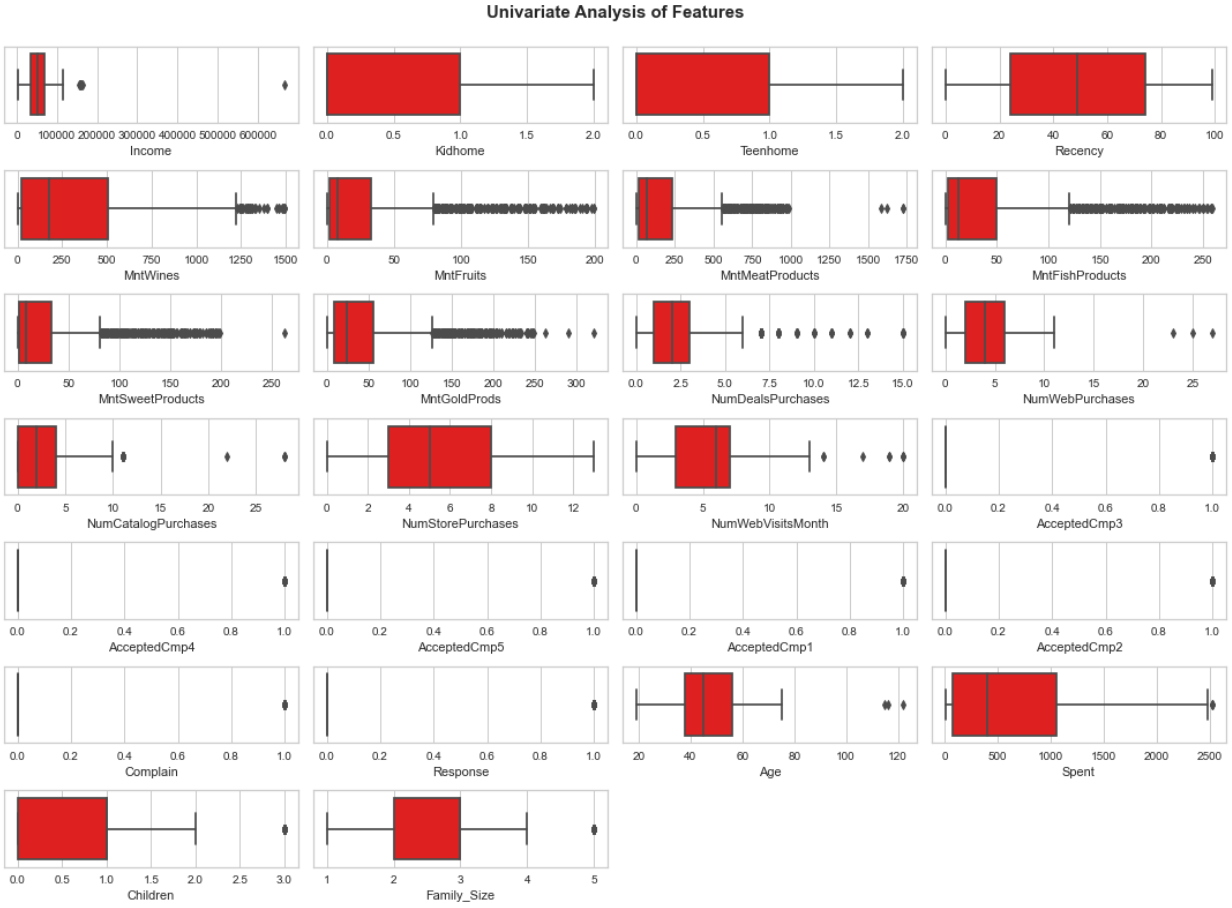
In [28]:
```
plt.figure(figsize=(15,15))
plt.suptitle('Univariate Analysis of Features',fontweight='bold',fontsize=15

for i in range(0,len(int_cols)):
    plt.subplot(10,4,i+1)
    sns.kdeplot(x=dataset[int_cols[i]],shade=True,color='red')
    plt.tight_layout()
```



In [29]:
```
plt.figure(figsize = (15,15))
plt.suptitle('Univariate Analysis of Features',fontweight='bold',fontsize=15

for i in range(0,len(int_cols)):
    plt.subplot(10,4,i+1)
    sns.boxplot(data=dataset,x=int_cols[i],color='red')
    plt.xlabel(int_cols[i])
    plt.tight_layout()
```



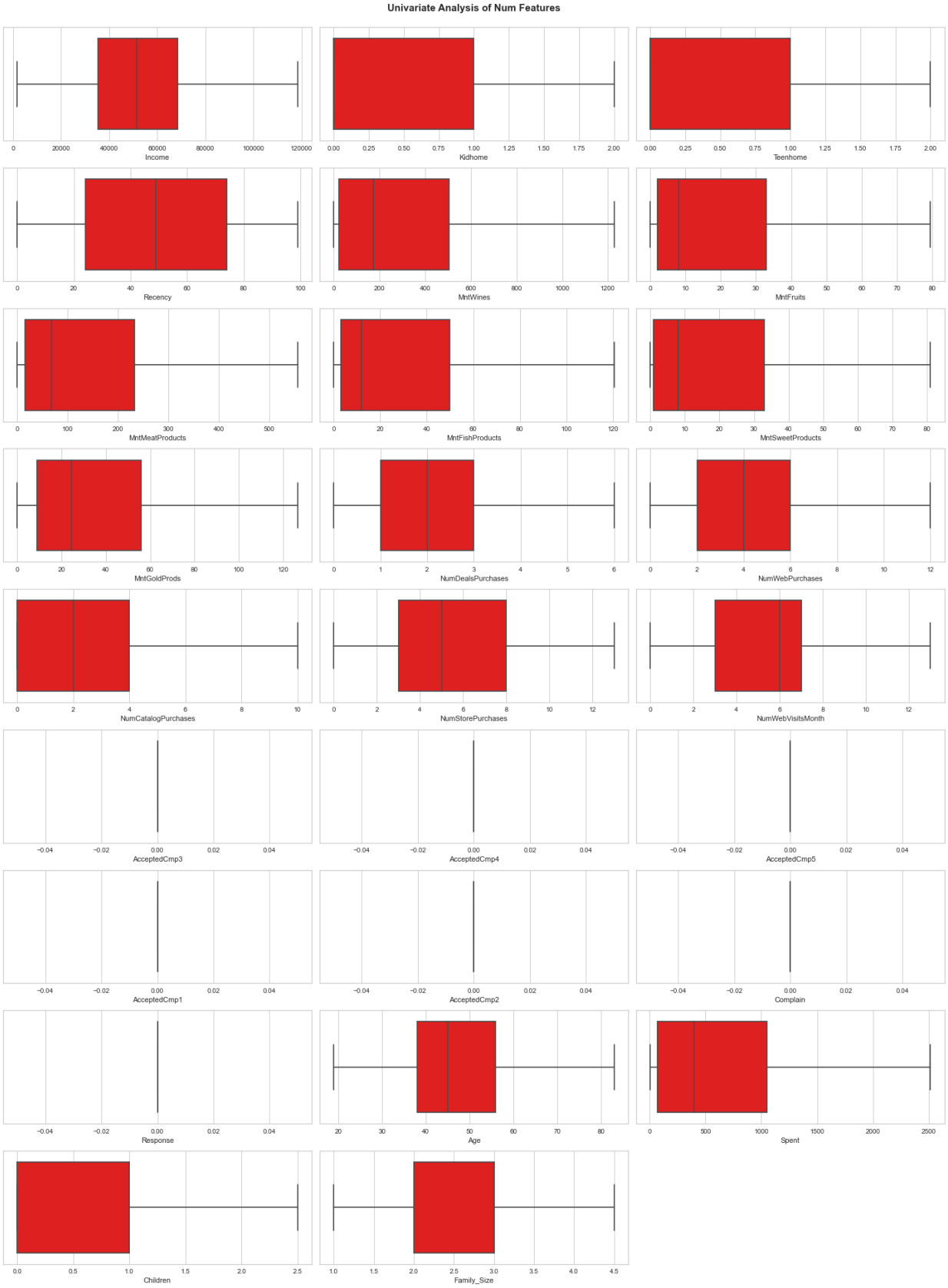
In [30]:
```
def remove_outliers(in_dataset, in_cols):

    first_quartile = in_dataset[in_cols].quantile(0.25)
    third_quartile = in_dataset[in_cols].quantile(0.75)
    iqr = third_quartile - first_quartile
    upper_limit = third_quartile + 1.5 * iqr
    lower_limit = first_quartile - 1.5 * iqr
    in_dataset.loc[(in_dataset[in_cols] > upper_limit), in_cols] = upper_limi
    in_dataset.loc[(in_dataset[in_cols] < lower_limit), in_cols] = lower_limi
    return in_dataset
```

In [31]:
```
for features in int_cols:
    dataset = remove_outliers(dataset,features)
```

In [32]:
```
plt.figure(figsize = (20,250))
plt.suptitle('Univariate Analysis of Num Features',fontweight='bold',fontsize

for i in range(0,len(int_cols)):
    plt.subplot(85,3,i+1)
    sns.boxplot(data=dataset,x=int_cols[i],color='red')
    plt.xlabel(int_cols[i])
    plt.tight_layout()
```



In [33]:
```
dataset['Education'] = dataset['Education'].map({'Undergraduate':0, 'Graduate
```

In [34]:
```
dataset['Living_With'] = dataset['Living_With'].map({'Alone':0, 'Partner':1})
```

In [35]:
```
dataset_ = dataset.copy()
```

In [36]:
```
cols = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'Acce
dataset = dataset.drop(cols, axis=1)
```

In [37]:
```
scaler = StandardScaler()
dataset = pd.DataFrame(scaler.fit_transform(dataset), columns = dataset.colur
```

In [38]:
```
p = PCA(n_components=3)
p.fit(dataset)
```

Out[38]:
```
PCA
PCA(n_components=3)
```

In [39]:
```
W = p.components_.T
W
```

Out[39]:
```
array([[ 0.01150469,  0.1264986 , -0.53007114],
       [ 0.27489741,  0.16292514, -0.11961512],
       [-0.23678125,  0.01225473,  0.26552554],
       [-0.09125218,  0.45338749, -0.18990445],
       [ 0.00411461,  0.01608669,  0.03336462],
       [ 0.25230411,  0.19924324, -0.09747587],
       [ 0.25300231,  0.02976789,  0.22022379],
       [ 0.29378553,  0.01341824,  0.04970482],
       [ 0.26073301,  0.00634505,  0.21130988],
       [ 0.25442806,  0.02980545,  0.22467603],
       [ 0.19807225,  0.13805676,  0.23164106],
       [-0.0858184 ,  0.37480305,  0.19986521],
       [ 0.17148314,  0.3072246 ,  0.10003722],
       [ 0.28029256,  0.09535581, -0.00716456],
       [ 0.23888557,  0.20335426,  0.01651975],
       [-0.22059467,  0.06867733,  0.1812646 ],
       [ 0.03910063,  0.21615014, -0.47996371],
       [ 0.30959299,  0.11968703,  0.02426995],
       [-0.0250982 ,  0.11981989,  0.2005631 ],
       [-0.23915457,  0.34003406,  0.05447765],
       [-0.20944241,  0.34321837,  0.15022368],
       [-0.23201991,  0.30311839,  0.0890357 ]])
```

In [40]:
```
p.explained_variance_
```

Out[40]:
```
array([8.84485331, 3.00939488, 1.4211903 ])
```

In [41]:
```
p.explained_variance_ratio_
```

Out[41]:
```
array([0.40185736, 0.13672895, 0.06457041])
```

In [42]:
```
p.explained_variance_ratio_.cumsum()
```

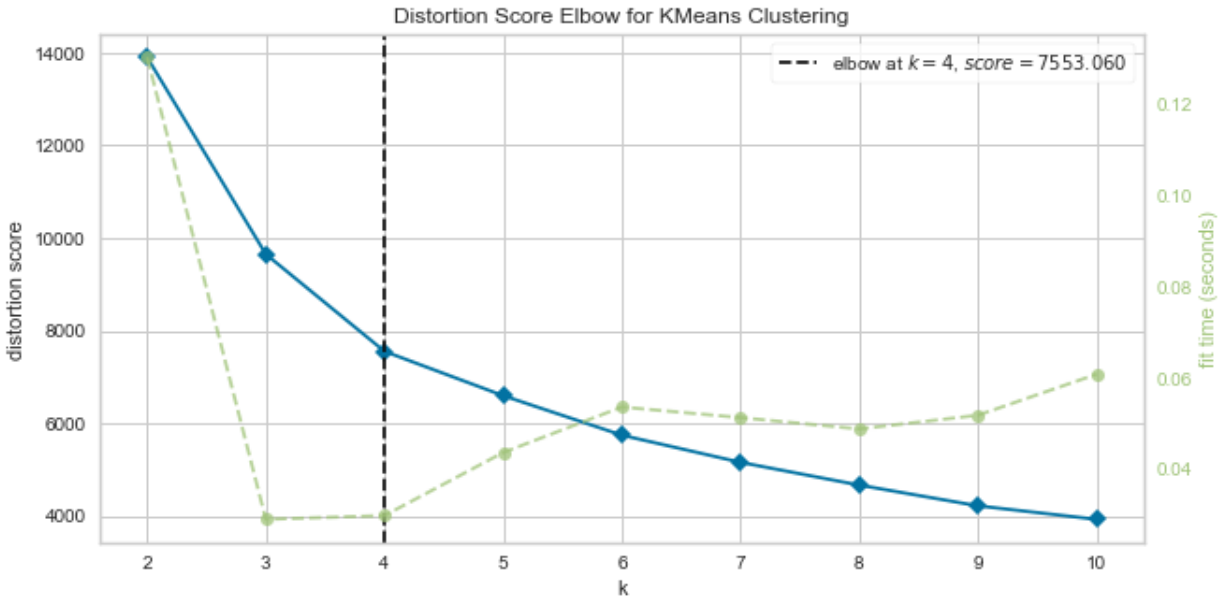
Out[42]:
```
array([0.40185736, 0.53858631, 0.60315672])
```

In [43]:
```
df_PCA = pd.DataFrame(p.transform(dataset), columns=(['col1', 'col2', 'col3']
df_PCA.describe()
```

Out[43]:

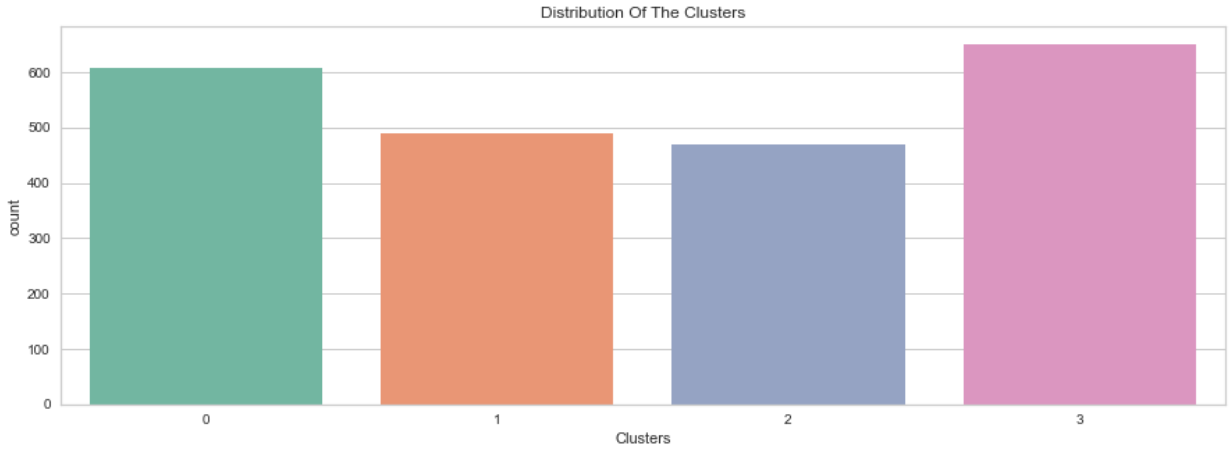
	col1	col2	col3
count	2.216000e+03	2.216000e+03	2.216000e+03
mean	2.308623e-16	-1.923852e-17	4.328667e-17
std	2.974030e+00	1.734761e+00	1.192137e+00
min	-5.240740e+00	-4.467226e+00	-3.523820e+00
25%	-2.669935e+00	-1.356149e+00	-8.438433e-01
50%	-8.383377e-01	-1.761448e-01	-1.938035e-02
75%	2.576374e+00	1.304850e+00	8.090423e-01
max	7.061699e+00	6.021683e+00	5.029512e+00

In [44]:
```
plt.figure(figsize=(10,5))
Elbow_M = KElbowVisualizer(KMeans(), k=10)
Elbow_M.fit(df_PCA)
Elbow_M.show()
plt.show()
```



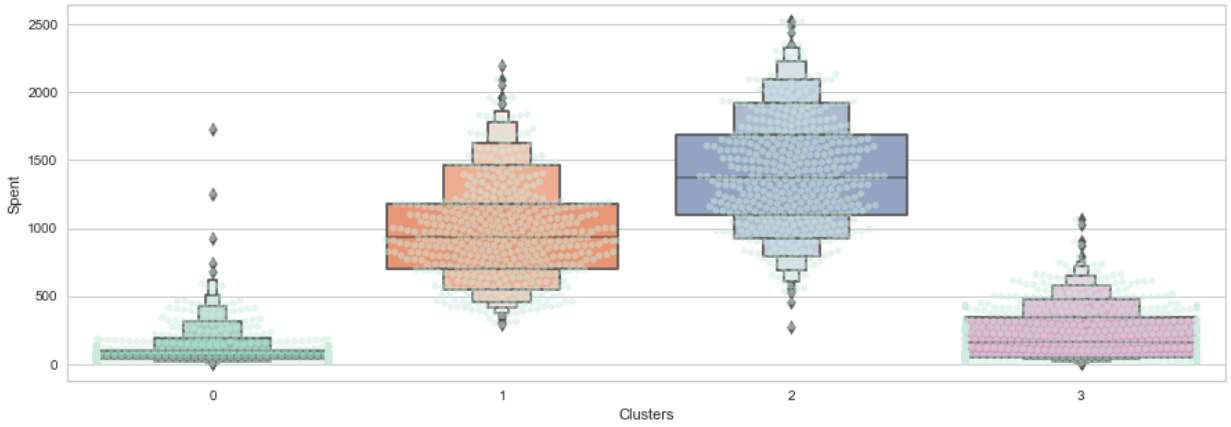
In [45]:
```
AC = AgglomerativeClustering(n_clusters=4)
yhat_AC = AC.fit_predict(df_PCA)
df_PCA['Clusters'] = yhat_AC
dataset['Clusters'] = yhat_AC
dataset_['Clusters'] = yhat_AC
```

In [46]:
```
plt.figure(figsize=(15,5))
pl = sns.countplot(x=dataset['Clusters'], palette='Set2')
pl.set_title('Distribution Of The Clusters')
plt.show()
```



In [47]:
```
plt.figure(figsize=(15,5))
pl = sns.scatterplot(data=dataset_, x=dataset_['Spent'], y=dataset_['Income']
pl.set_title('Clusters Profile Based on Income and Spending')
plt.legend()
plt.show()
```

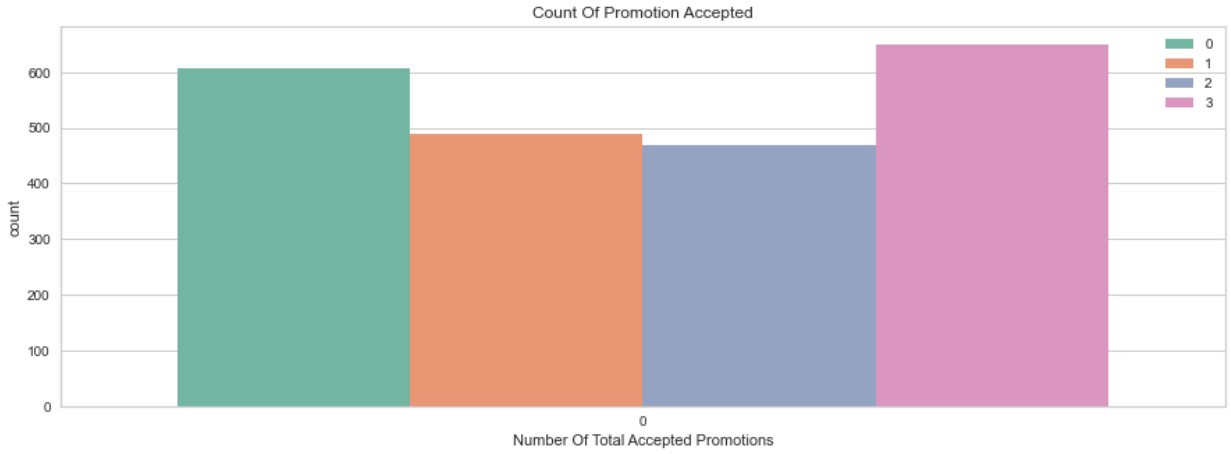
In [48]:
```
plt.figure(figsize=(15,5))
pl = sns.swarmplot(x=dataset_['Clusters'], y=dataset_['Spent'], color='#CBEDD
pl = sns.boxenplot(x=dataset_['Clusters'], y=dataset_['Spent'], palette='Set2
plt.show()
```



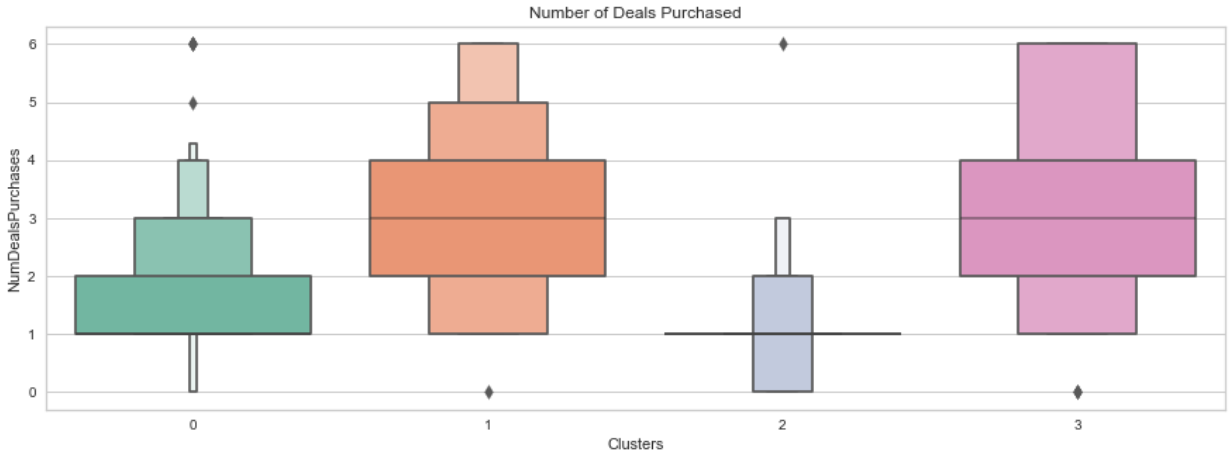
In [51]:
```
dataset_['Total_Promos'] = dataset_['AcceptedCmp1'] + dataset_['AcceptedCmp2

plt.figure(figsize=(15,5))
pl = sns.countplot(x=dataset_['Total_Promos'], hue=dataset_['Clusters'], pale
```

```
pl.set_title('Count Of Promotion Accepted')
pl.set_xlabel('Number Of Total Accepted Promotions')
plt.legend(loc='upper right')
plt.show()
```



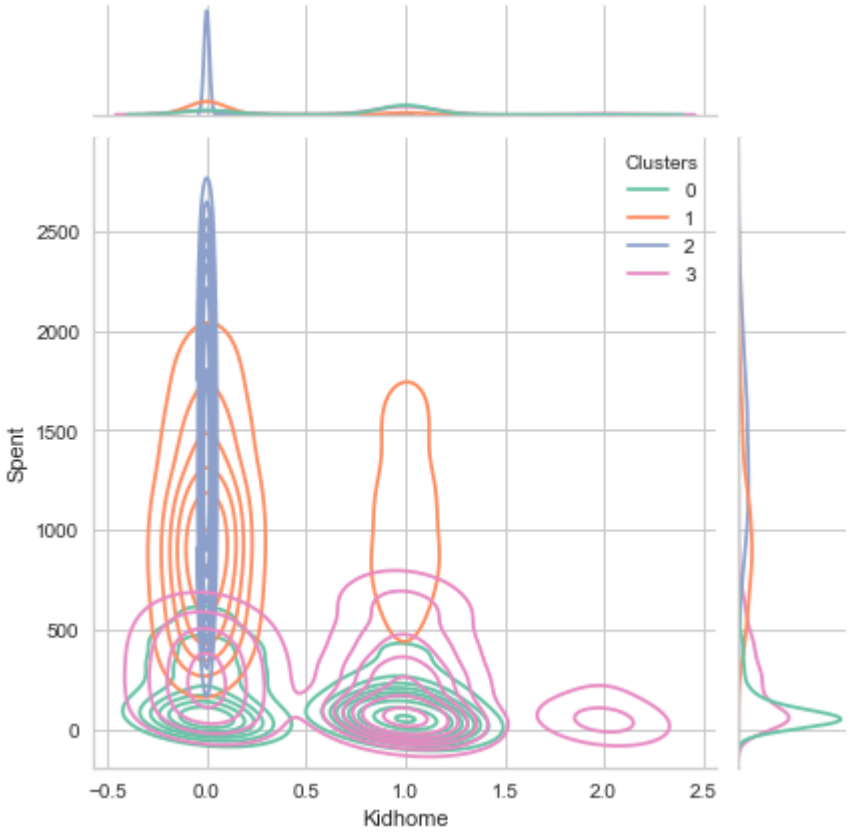
In [52]:
```
plt.figure(figsize=(15,5))
pl = sns.boxenplot(y=dataset_['NumDealsPurchases'], x=dataset_['Clusters'], p
pl.set_title('Number of Deals Purchased')
plt.show()
```



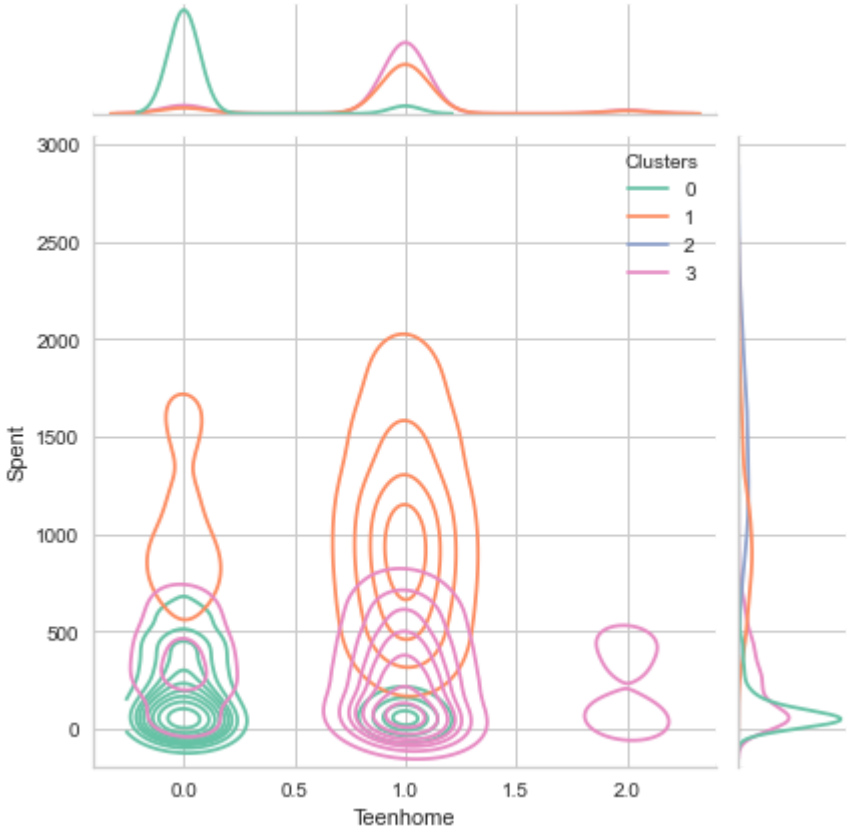
In [55]:
```
cols = ['Kidhome', 'Teenhome', 'Age', 'Children', 'Family_Size', 'Is_Parent'

for i in cols:
    plt.figure(figsize=(15,5))
    sns.jointplot(x=dataset_[i], y=dataset_['Spent'], hue=dataset_['Clusters
plt.show()
```

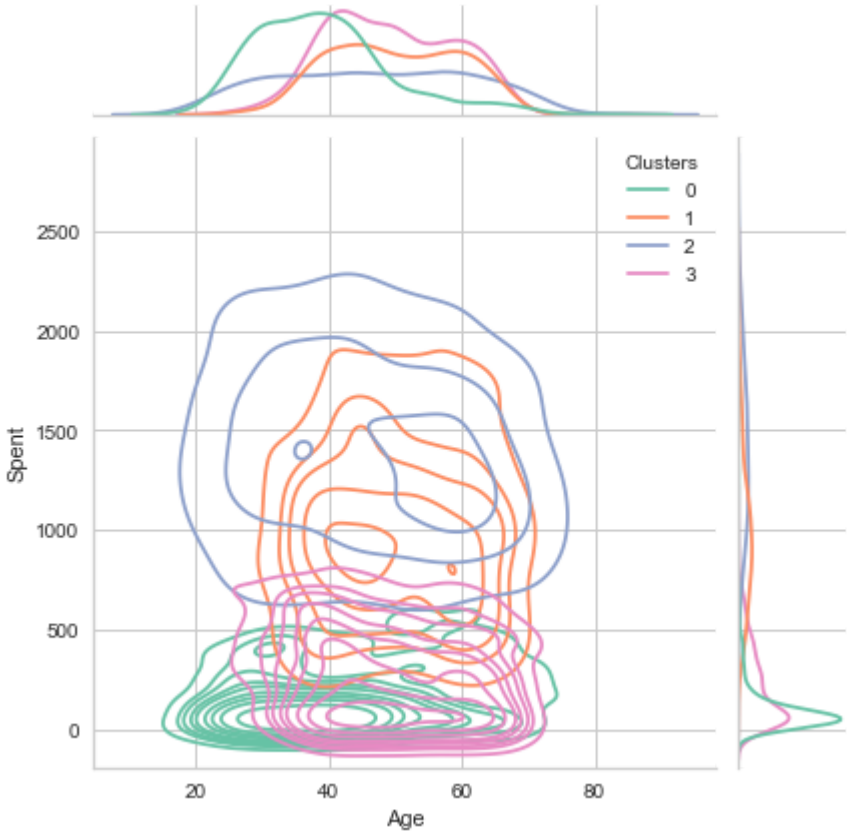
```
<Figure size 1080x360 with 0 Axes>
```



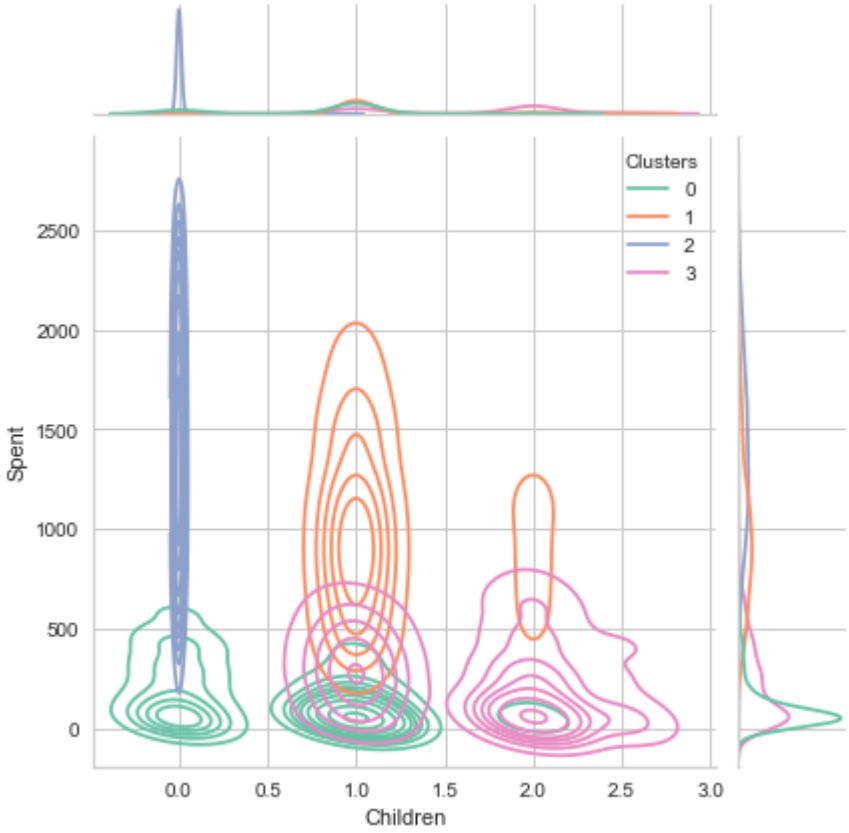
<Figure size 1080x360 with 0 Axes>



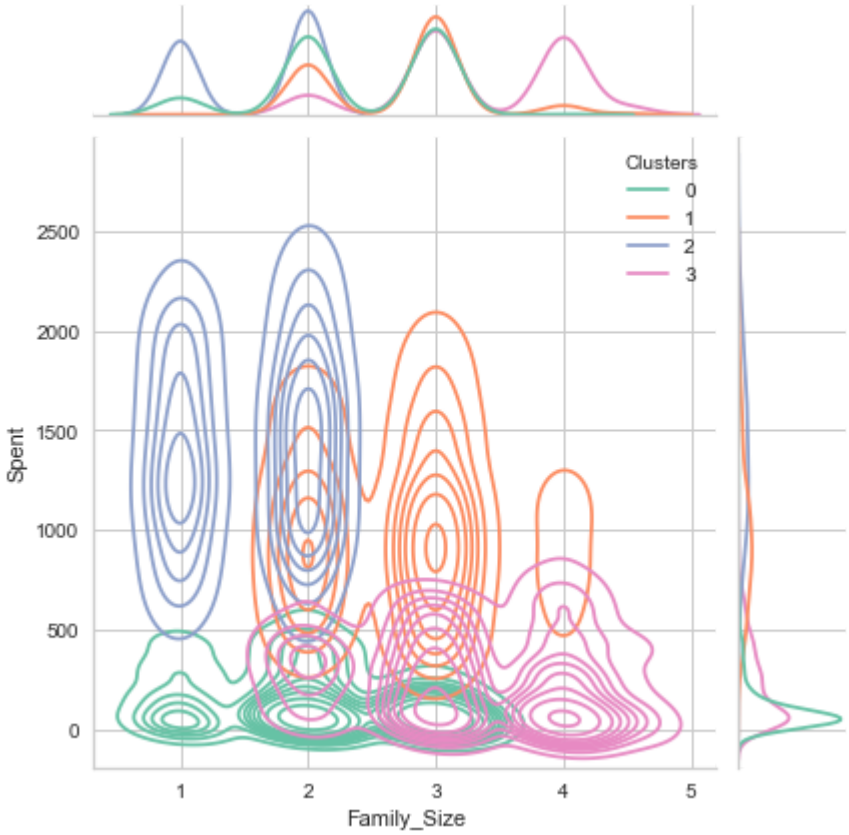
<Figure size 1080x360 with 0 Axes>



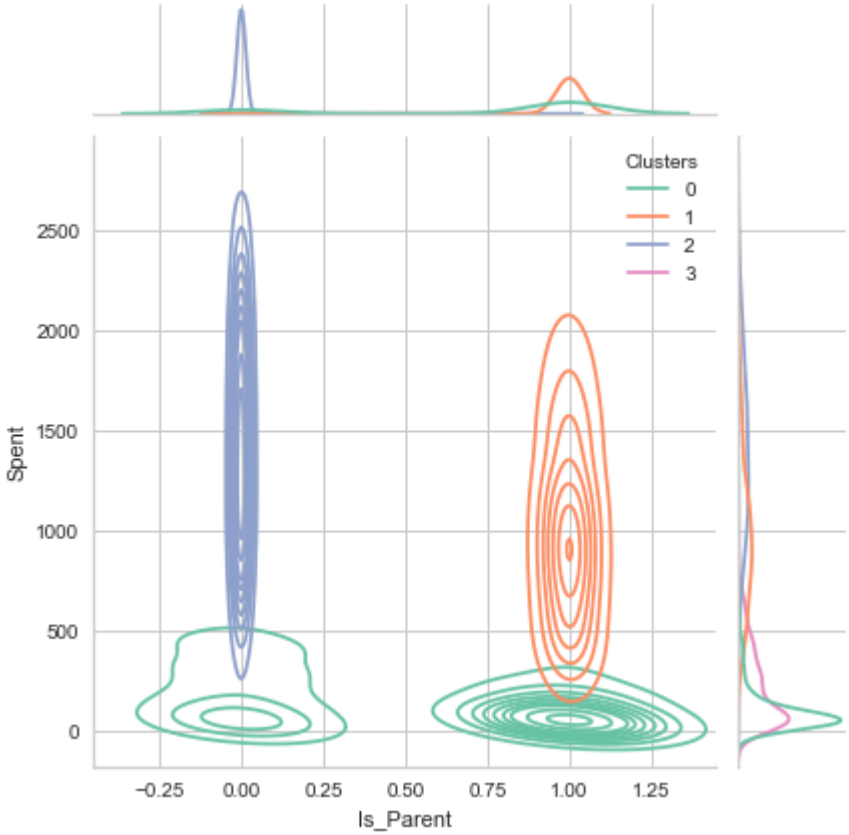
<Figure size 1080x360 with 0 Axes>



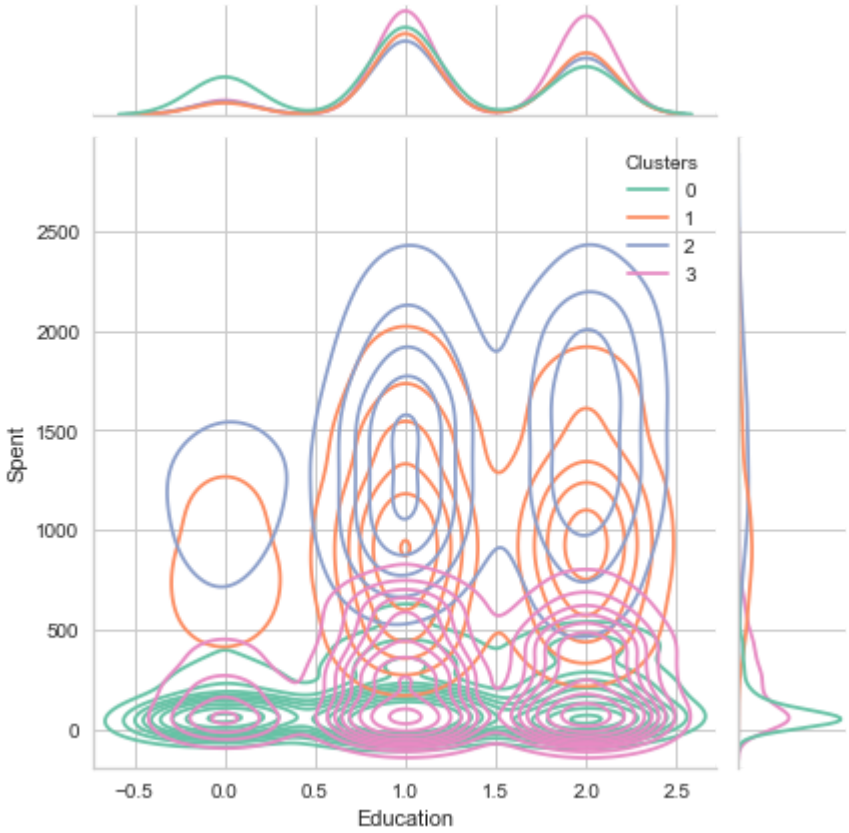
<Figure size 1080x360 with 0 Axes>



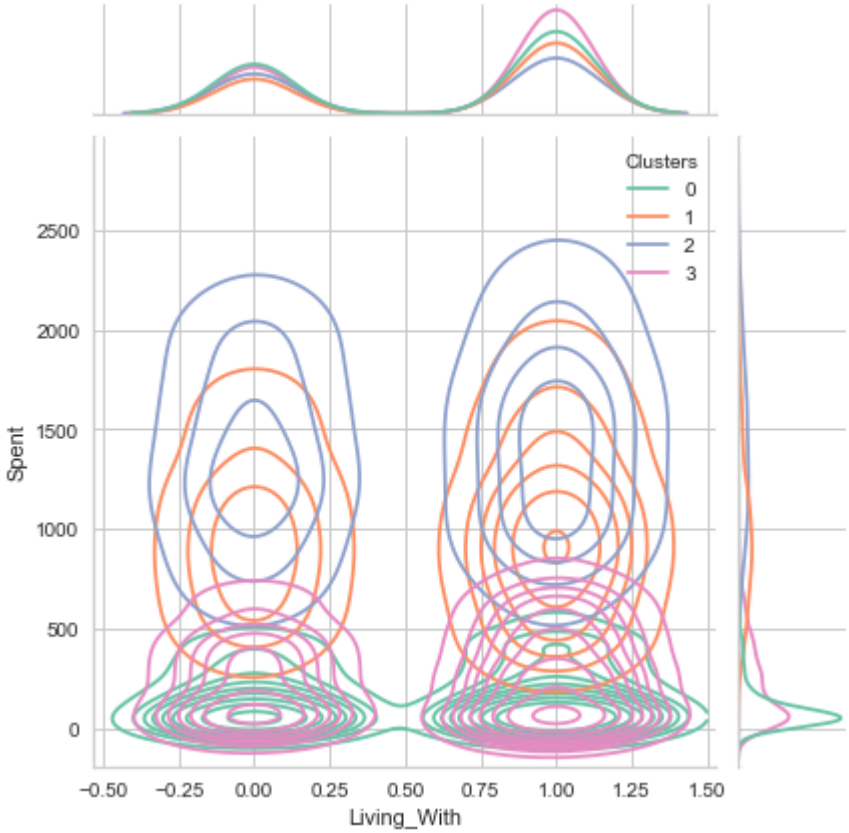
<Figure size 1080x360 with 0 Axes>



<Figure size 1080x360 with 0 Axes>




```
<Figure size 1080x360 with 0 Axes>
```



In [62]:
```
dataset_ = dataset_.astype(int)
```

In [58]:
```
kmeans_model = KMeans(4)
kmeans_model.fit_predict(dataset_)
```

Out[58]:
```
array([3, 0, 1, ..., 3, 3, 3])
```

In [67]:
```
pca_df_kmeans = pd.concat([dataset_,pd.DataFrame({'Clusters':kmeans_model.lab
```

In [69]:
```
X = dataset_.drop(['Clusters'], axis=1)
y = dataset_[['Clusters']]
X_train, X_test, y_train, y_test =train_test_split(X, y, test_size=0.2)
```

In [75]:
```
dt = DecisionTreeClassifier(criterion='entropy')
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)

print(metrics.confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[ 99   0   3   5]
 [  2  85   1   8]
 [  0   0  90   0]
 [  5   9   0 137]]
              precision    recall  f1-score   support

           0       0.93      0.93      0.93       107
           1       0.90      0.89      0.89        96
           2       0.96      1.00      0.98        90
           3       0.91      0.91      0.91       151

    accuracy                           0.93       444
   macro avg       0.93      0.93      0.93       444
weighted avg       0.93      0.93      0.93       444
```

In [74]:
```
rf = RandomForestClassifier(criterion='entropy')
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)

print(metrics.confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[ 97   1   2   7]
 [  0  92   1   3]
 [  1   0  89   0]
 [  4   6   0 141]]
              precision    recall  f1-score   support

           0       0.95      0.91      0.93       107
           1       0.93      0.96      0.94        96
           2       0.97      0.99      0.98        90
           3       0.93      0.93      0.93       151

    accuracy                           0.94       444
   macro avg       0.95      0.95      0.95       444
weighted avg       0.94      0.94      0.94       444
```

In [76]:
```
xgb = XGBClassifier(criterion='entropy')
xgb.fit(X_train, y_train)
y_pred = xgb.predict(X_test)
```

```
print(metrics.confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[09:53:58] WARNING: C:/Users/administrator/workspace/xgboost-win64_release_
1.6.0/src/learner.cc:627:
Parameters: { "criterion" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actua
lly being used
  but getting flagged wrongly here. Please open an issue if you find any suc
h cases.


[[ 98   1   2   6]
 [  0  91   1   4]
 [  1   0  89   0]
 [  4   4   0 143]]
              precision    recall  f1-score   support

           0       0.95      0.92      0.93       107
           1       0.95      0.95      0.95        96
           2       0.97      0.99      0.98        90
           3       0.93      0.95      0.94       151

    accuracy                           0.95       444
   macro avg       0.95      0.95      0.95       444
weighted avg       0.95      0.95      0.95       444
```

In [78]:
```
dataset_.to_csv('Clustered_Customer_Data.csv')
```

In [80]:
```
filename = 'model.pkl'
pickle.dump(xgb, open(filename, 'wb'))

loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test, y_test)
print(result,'% Acuuracy')
```

```
0.9481981981981982 % Acuuracy
```

In [85]:
```
# Thank You!
```