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.

Column Information People 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
```

```
Target
Need to perform clustering to summarize customer segments.
```

In [1]:

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

In [2]:

```
#READ THE DATASET...
df = pd.read_csv("C:\\Users\\sneha\\Downloads\\marketing_campaign.csv", sep
```

In [3]:

```
df.head()
```

Out[3]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer
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

5 rows × 29 columns

In [123]:
```
df.columns
```

Out[123]:
```
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 [124]:
```
df.shape
```

Out[124]:
```
(2240, 29)
```

In [125]:
```
df.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 [126]:
```
df.describe().T
```

Out[126]:

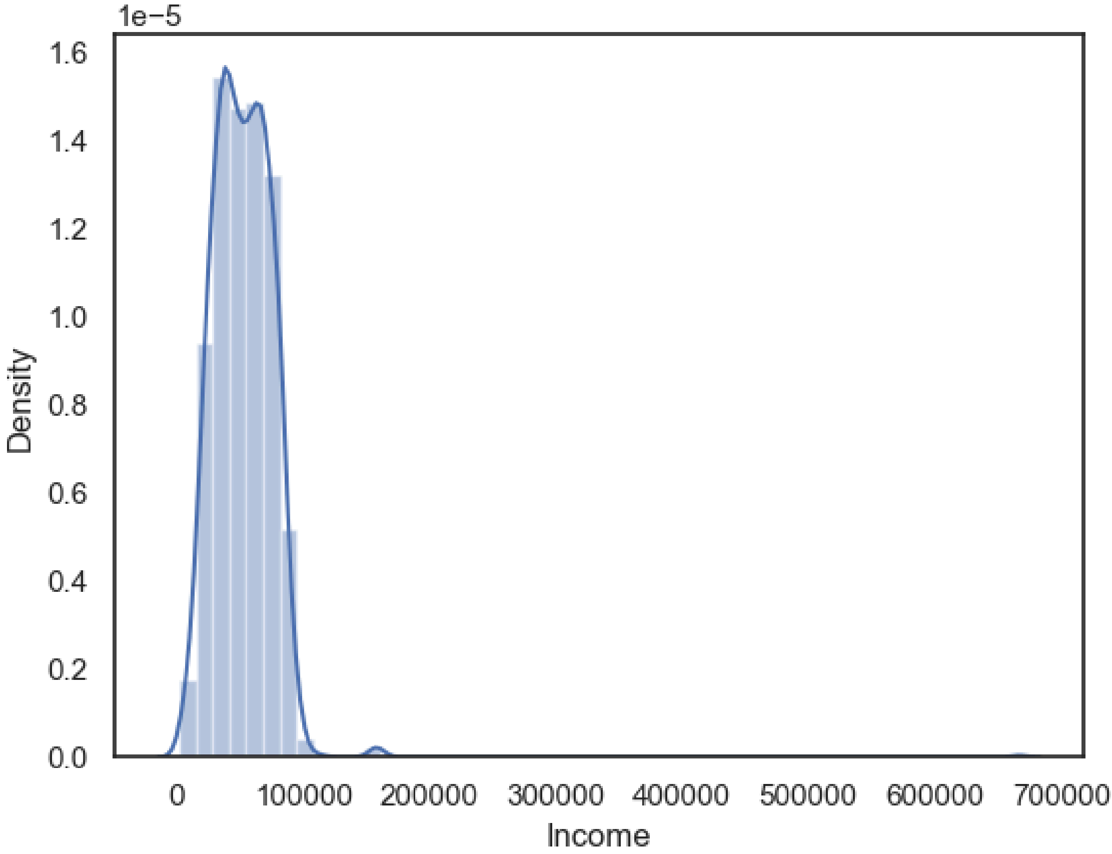
	count	mean	std	min	25%	50%	7!
ID	2240.0	5592.159821	3246.662198	0.0	2828.25	5458.5	8427.
Year_Birth	2240.0	1968.805804	11.984069	1893.0	1959.00	1970.0	1977.
Income	2216.0	52247.251354	25173.076661	1730.0	35303.00	51381.5	68522.
Kidhome	2240.0	0.444196	0.538398	0.0	0.00	0.0	1.
Teenhome	2240.0	0.506250	0.544538	0.0	0.00	0.0	1.
Recency	2240.0	49.109375	28.962453	0.0	24.00	49.0	74.
MntWines	2240.0	303.935714	336.597393	0.0	23.75	173.5	504.
MntFruits	2240.0	26.302232	39.773434	0.0	1.00	8.0	33.
MntMeatProducts	2240.0	166.950000	225.715373	0.0	16.00	67.0	232.
MntFishProducts	2240.0	37.525446	54.628979	0.0	3.00	12.0	50.
MntSweetProducts	2240.0	27.062946	41.280498	0.0	1.00	8.0	33.
MntGoldProds	2240.0	44.021875	52.167439	0.0	9.00	24.0	56.
NumDealsPurchases	2240.0	2.325000	1.932238	0.0	1.00	2.0	3.
NumWebPurchases	2240.0	4.084821	2.778714	0.0	2.00	4.0	6.
NumCatalogPurchases	2240.0	2.662054	2.923101	0.0	0.00	2.0	4.
NumStorePurchases	2240.0	5.790179	3.250958	0.0	3.00	5.0	8.
NumWebVisitsMonth	2240.0	5.316518	2.426645	0.0	3.00	6.0	7.
AcceptedCmp3	2240.0	0.072768	0.259813	0.0	0.00	0.0	0.
AcceptedCmp4	2240.0	0.074554	0.262728	0.0	0.00	0.0	0.
AcceptedCmp5	2240.0	0.072768	0.259813	0.0	0.00	0.0	0.
AcceptedCmp1	2240.0	0.064286	0.245316	0.0	0.00	0.0	0.
AcceptedCmp2	2240.0	0.013393	0.114976	0.0	0.00	0.0	0.
Complain	2240.0	0.009375	0.096391	0.0	0.00	0.0	0.
Z_CostContact	2240.0	3.000000	0.000000	3.0	3.00	3.0	3.
Z_Revenue	2240.0	11.000000	0.000000	11.0	11.00	11.0	11.
Response	2240.0	0.149107	0.356274	0.0	0.00	0.0	0.

In [127]:
```
df.isna().sum()
```

Out[127]:
```
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
```

since there are some missing values in Income we will check that column and replace missing values with mean or median

In [128]:
```
sns.distplot(df['Income'])
plt.show()
```



since the data is left skewed we will replace the missing values with median

In [129]:
```
#FILL THE MISSING VALUES WITH THE MEDIAN VALUES..
df['Income']=df['Income'].fillna(df['Income'].median())
```

In [130]:
```
df[df.duplicated()]
```

Out[130]:

ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Rec

0 rows × 29 columns

In [131]:
```
#FINDING THE NUMBER OF UNIQUE VALUES PRESENT IN EACH COLUMN...
df.nunique()
```

Out[131]:
```
ID                     2240
Year_Birth               59
Education                 5
Marital_Status            8
Income                 1975
Kidhome                   3
Teenhome                  3
Dt_Customer             663
Recency                 100
MntWines                776
MntFruits               158
MntMeatProducts         558
MntFishProducts         182
MntSweetProducts        177
MntGoldProds            213
NumDealsPurchases        15
NumWebPurchases          15
NumCatalogPurchases      14
NumStorePurchases        14
NumWebVisitsMonth        16
AcceptedCmp3              2
AcceptedCmp4              2
AcceptedCmp5              2
AcceptedCmp1              2
AcceptedCmp2              2
Complain                  2
Z_CostContact             1
Z_Revenue                 1
Response                  2
dtype: int64
```

Note:-In above cell "Z_CostContact" and "Z_Revenue" have same value in all the rows that's why , they are not going to contribute anything in the model building. So we can drop them.

In [132]:
```
df=df.drop(columns=["Z_CostContact", "Z_Revenue"],axis=1)
```

Univariate Analysis :-

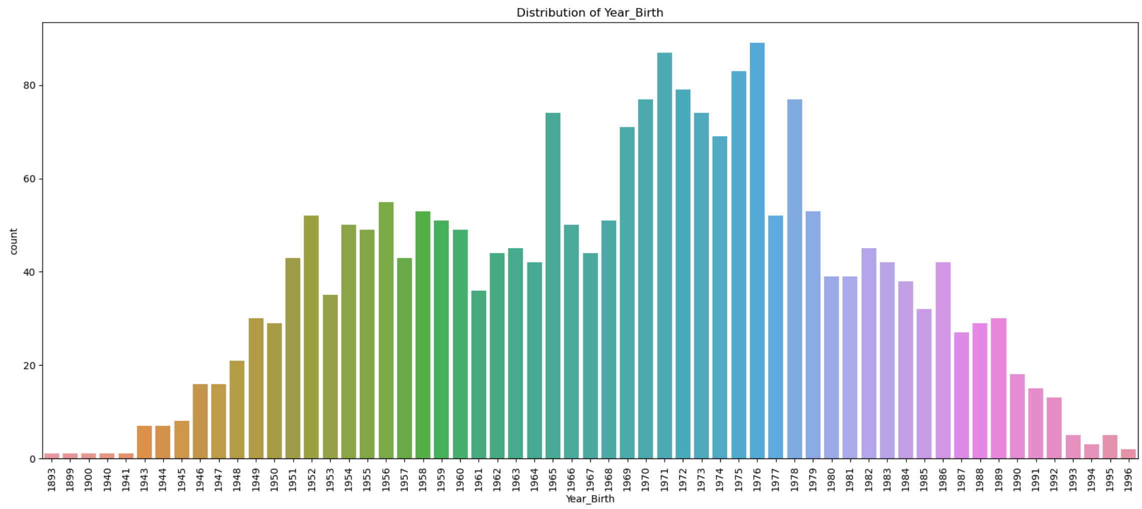
1.Analysis on Year_Birth Variable.

In [133]:
```
#CHECKING NUMBER OF UNIQUE CATEGORIES PRESENT IN THE "Year_Birth"
print("Unique categories present in the Year_Birth:",df["Year_Birth"].value
```

```
Unique categories present in the Year_Birth: 1976    89
1971    87
1975    83
1972    79
1978    77
1970    77
1973    74
1965    74
1969    71
1974    69
1956    55
1958    53
1979    53
1952    52
1977    52
1968    51
1959    51
1966    50
1954    50
1955    49
1960    49
1982    45
1963    45
1967    44
1962    44
1957    43
1951    43
1983    42
1986    42
1964    42
1980    39
1981    39
1984    38
1961    36
1953    35
1985    32
1989    30
1949    30
1950    29
1988    29
1987    27
1948    21
1990    18
1946    16
1947    16
1991    15
1992    13
1945     8
1943     7
1944     7
1993     5
1995     5
1994     3
1996     2
1899     1
1941     1
1893     1
1900     1
1940     1
Name: Year_Birth, dtype: int64
```

In [6]:
```
import matplotlib.pyplot as plt
import seaborn as sns

def uni_V(df, col):
    # Check for null values and data type
    if df[col].isnull().sum() > 0:
        print(f"Column '{col}' contains null values. Please handle them bef
        return
    if not pd.api.types.is_numeric_dtype(df[col]):
        print(f"Column '{col}' should contain numeric values. Please check
        return

    plt.figure(figsize=(20, 8))
    sns.countplot(x=df[col])
    plt.xticks(rotation=90)
    plt.title(f'Distribution of {col}')
    plt.show()
```

In [8]:
```
uni_V(df,'Year_Birth')
```



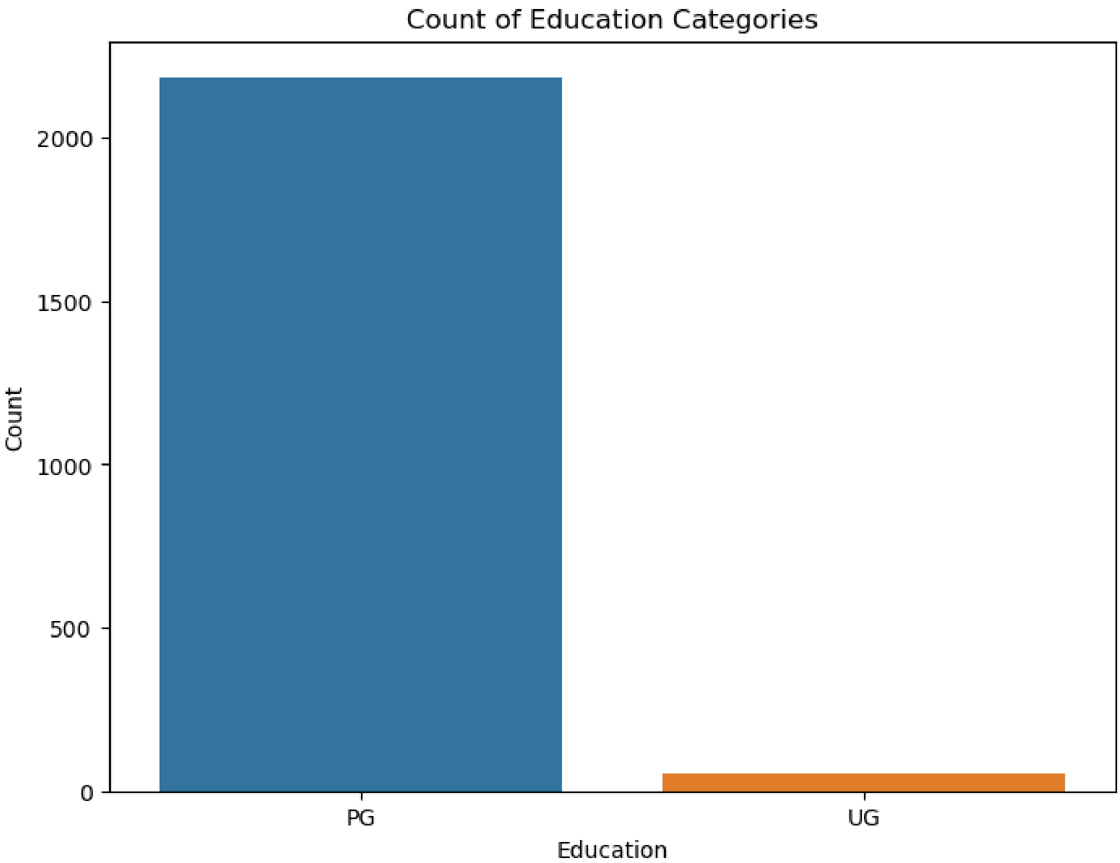
Data points in year birth are uniformly distributed

2.Analysis On Education Variable.

In [9]:
```
df['Education'].unique()
```

Out[9]:
```
array(['Graduation', 'PhD', 'Master', 'Basic', '2n Cycle'], dtype=object)
```

In [10]:
```
def uni_V(col):
    plt.figure(figsize=(8,6))
    sns.countplot(x=col, data=df)
    plt.xlabel('Education')
    plt.ylabel('Count')
    plt.title('Count of Education Categories')
    plt.show()

# Changing category into "UG" and "PG" only
df['Education'] = df['Education'].replace(['PhD', '2n Cycle', 'Graduation',
df['Education'] = df['Education'].replace(['Basic'], 'UG')

# Plotting
uni_V('Education')
```



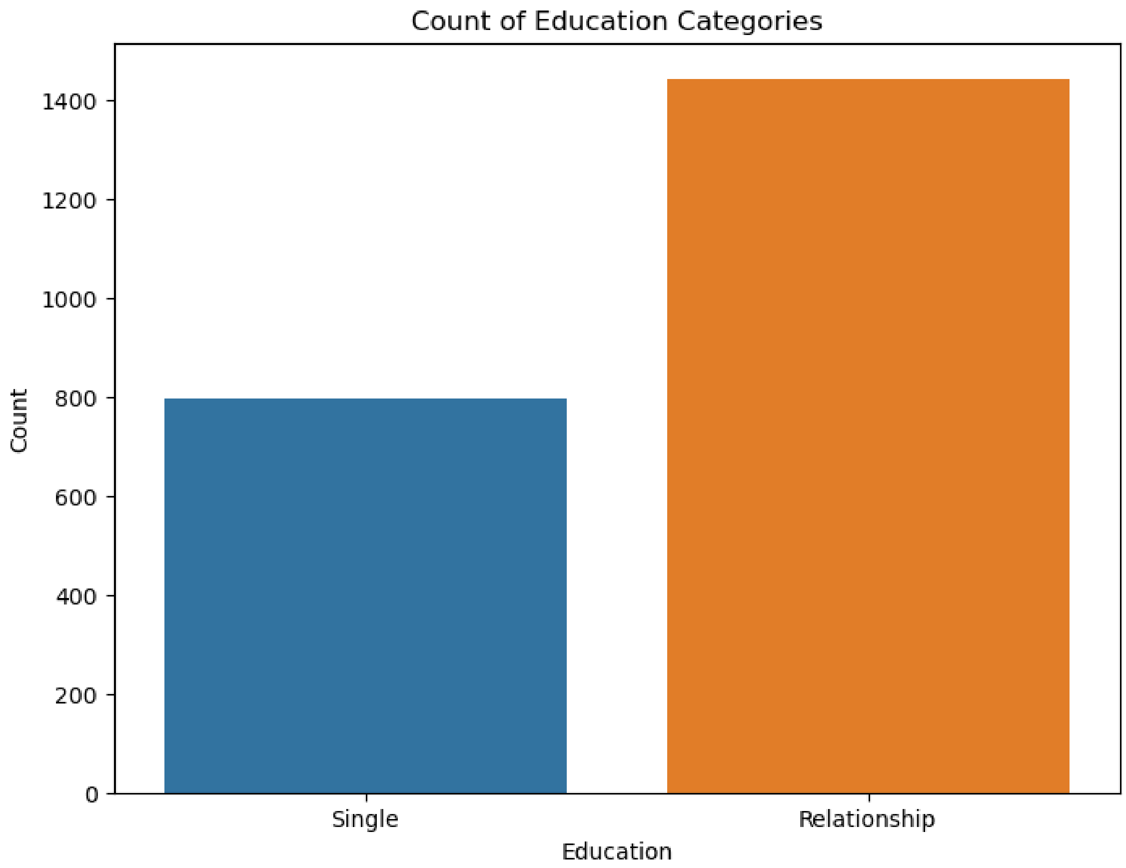
We observed that most of the data points here are post-Graduated

3.Analysis On Marital_Status Variable.

In [31]:
```
df['Marital_Status'].unique()
```

Out[31]:
```
array(['Single', 'Together', 'Married', 'Divorced', 'Widow', 'Alone',
       'Absurd', 'YOLO'], dtype=object)
```

In [32]:
```
#REPLACING THE CONFLICT VALUES IN Marital_status..
df['Marital_Status'] = df['Marital_Status'].replace(['Married', 'Together']
df['Marital_Status'] = df['Marital_Status'].replace(['Divorced', 'Widow', '
uni_V('Marital_Status')
```



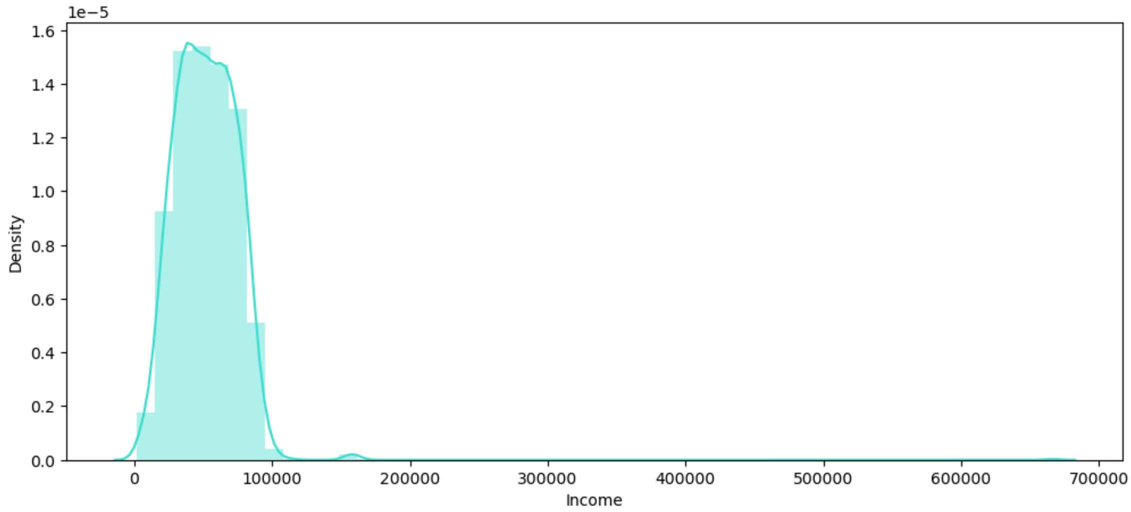
64.46% of Customers in the dataset are in "Relationship". 35.53% of Customers in the dataset are "Single".

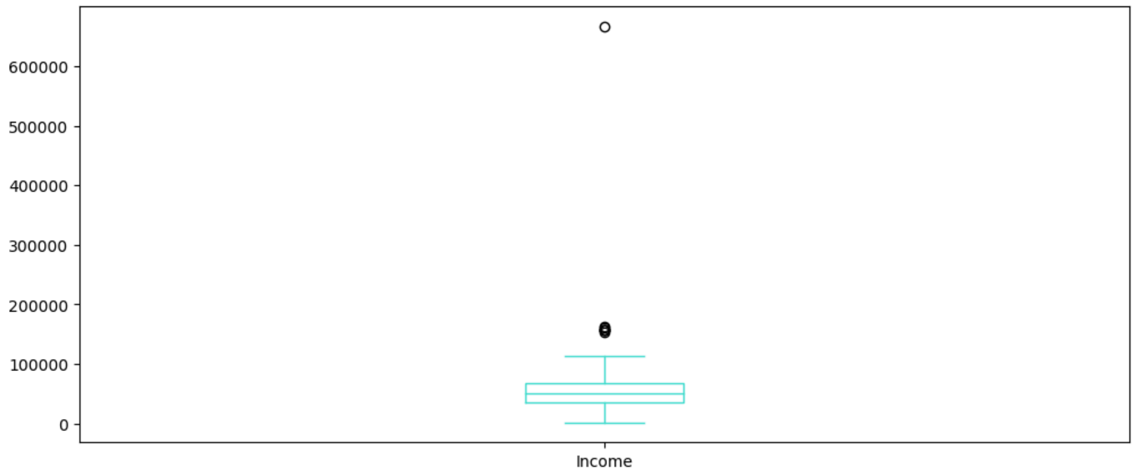
4.Analysis On Income Variable.

In [33]:
```
df['Income'].describe()
```

Out[33]:
```
count      2240.000000
mean      52237.975446
std       25037.955891
min        1730.000000
25%       35538.750000
50%       51381.500000
75%       68289.750000
max      666666.000000
Name: Income, dtype: float64
```

In [34]:
```
plt.figure(figsize=(12,5))
sns.distplot(df["Income"],color = 'turquoise')
plt.show()
df["Income"].plot.box(figsize=(12,5),color = 'turquoise')
plt.show()
```





The income column is left skewed as we saw earrlier but it has some outliers that we will treat it in later stage while model building

5.Analysis On "Kidhome,Teenhome" Variable.

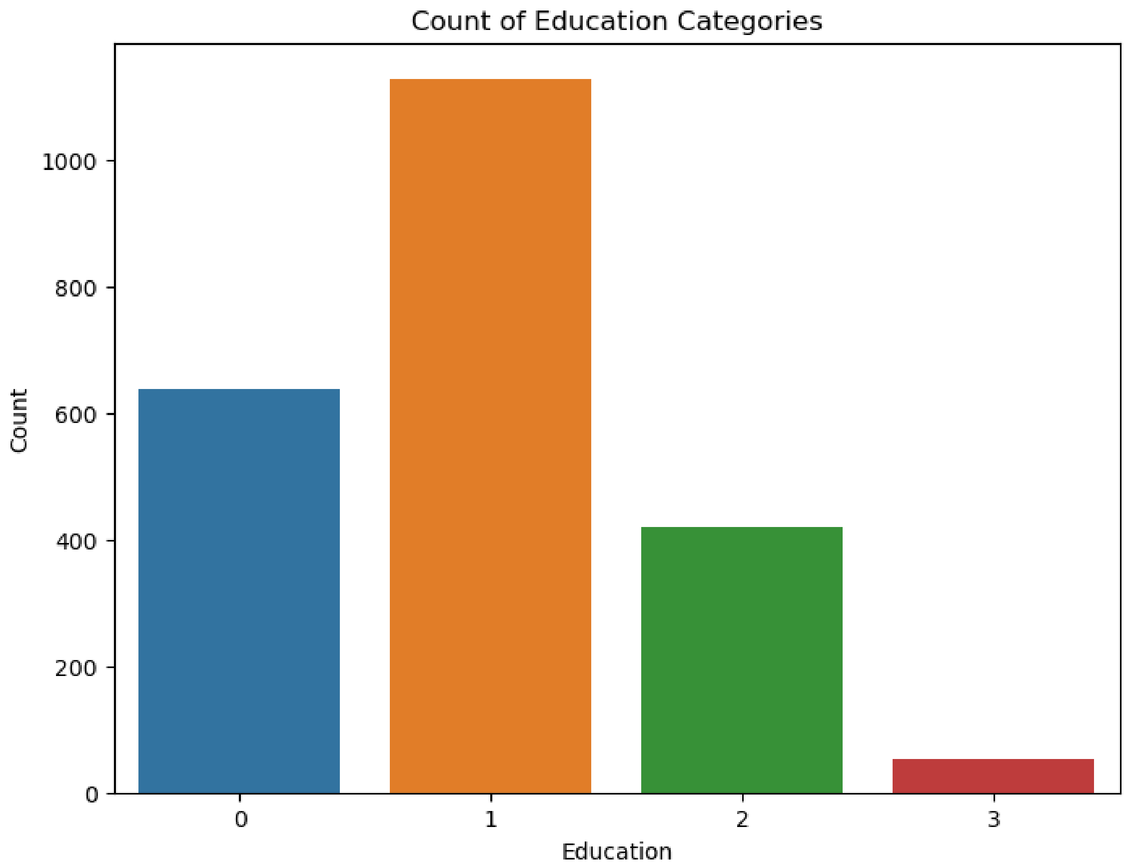
In [35]:
```
df['Teenhome'].unique()
```

Out[35]:
```
array([0, 1, 2], dtype=int64)
```

In [36]:
```
df['Kidhome'].unique()
```

Out[36]:
```
array([0, 1, 2], dtype=int64)
```

In [38]:
```
# Combining different dataframe into a single column to reduce the number o

df['Kids'] = df['Kidhome'] + df['Teenhome']
uni_V('Kids')
```



50.35% of Customers in the dataset have 1 kid. 28.48% of Customers in the dataset have no kids. 18.79% of Customers in the dataset have 2 kids. 2.36% of Customers in the dataset have 3 kids.

6.Analysis On "MntWines,MntMeatProducts,MntFishProducts,MntSweetProducts,MntGoldProds" Variable.

In [39]:
```
df[['MntFruits','MntMeatProducts']].head()
```

Out[39]:

	MntFruits	MntMeatProducts
0	88	546
1	1	6
2	49	127
3	4	20
4	43	118

In [40]:
```
df['MntFishProducts'].nunique()
```

Out[40]:
```
182
```

In [41]:
```
df['MntFruits'].nunique()
```

Out[41]:
```
158
```

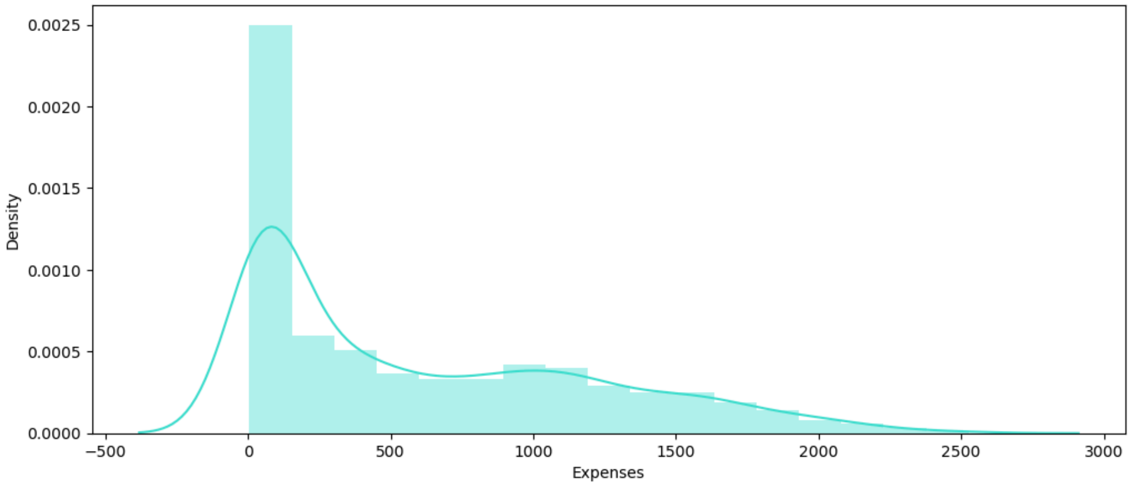
In [42]:
```
# Combining different dataframe into a single column to reduce the number o

df['Expenses'] = df['MntWines'] + df['MntFruits'] + df['MntMeatProducts'] +
df['Expenses'].head(10)
```

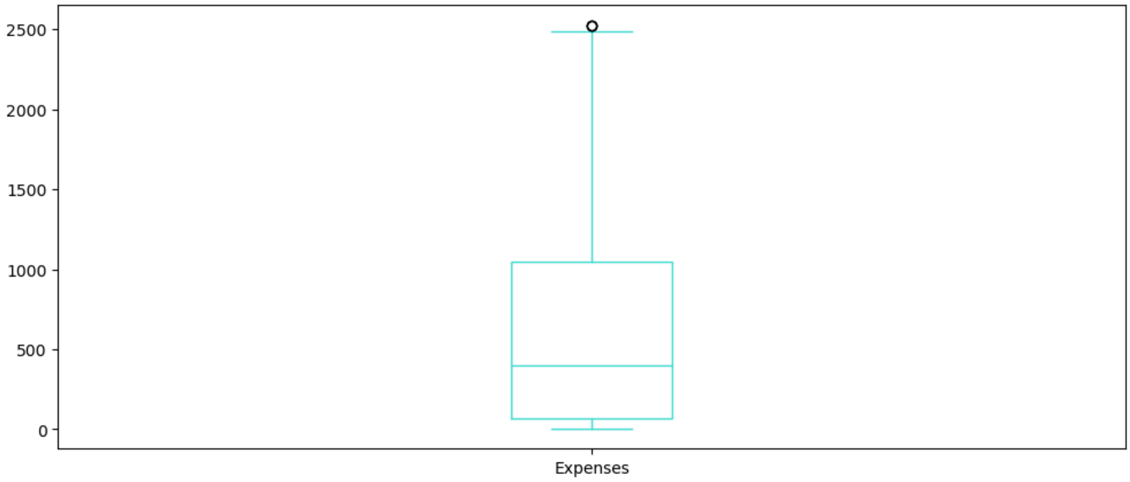
Out[42]:
```
0    1617
1      27
2     776
3      53
4     422
5     716
6     590
7     169
8      46
9      49
Name: Expenses, dtype: int64
```

In [43]:
```
df['Expenses'].describe()
```

Out[43]:
```
count    2240.000000
mean      605.798214
std       602.249288
min         5.000000
25%        68.750000
50%       396.000000
75%      1045.500000
max      2525.000000
Name: Expenses, dtype: float64
```

In [44]:
```
plt.figure(figsize=(12,5))
sns.distplot(df["Expenses"],color = 'turquoise')
plt.show()
df["Expenses"].plot.box(figsize=(12,5),color='turquoise')
plt.show()
```





The distribution of expenses is uniform

7.Analysis on "AcceptedCmp1,AcceptedCmp2,AcceptedCmp3,AcceptedCmp4,AcceptedCmp5" Variable.

In [45]:
```
df['AcceptedCmp1'].unique()
```

Out[45]: array([0, 1], dtype=int64)

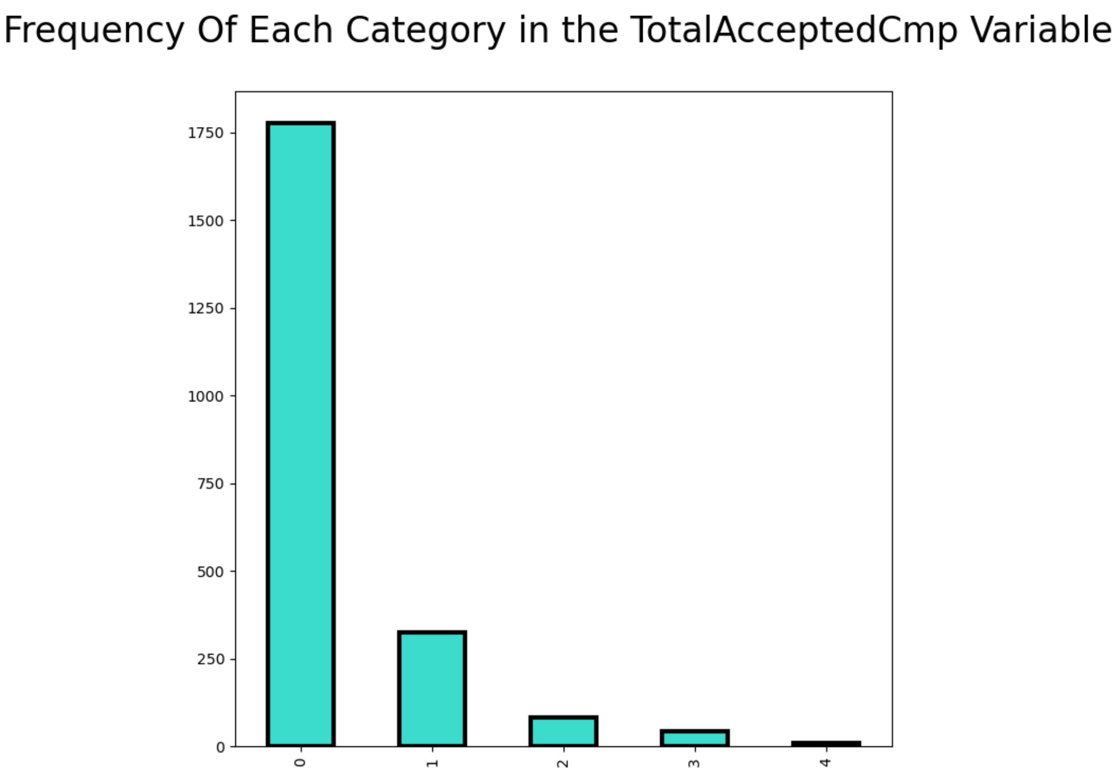
In [46]:
```
df['AcceptedCmp2'].unique()
```

Out[46]: array([0, 1], dtype=int64)

In [47]:
```
df['TotalAcceptedCmp'] = df['AcceptedCmp1'] + df['AcceptedCmp2'] + df['Acce
```

In [48]:
```
#CHECKING NUMBER OF UNIQUE CATEGORIES PRESENT IN THE "TotalAcceptedCmp"
print("Unique categories present in the TotalAcceptedCmp:",df['TotalAccepte
print("\n")

#VISUALIZING THE "TotalAcceptedCmp"


plt.figure(figsize=(8,8))
df['TotalAcceptedCmp'].value_counts().plot(kind='bar',color = 'turquoise',e
plt.title("Frequency Of Each Category in the TotalAcceptedCmp Variable \n",
plt.show()
```

```
Unique categories present in the TotalAcceptedCmp: 0    1777
1     325
2      83
3      44
4      11
Name: TotalAcceptedCmp, dtype: int64
```



79.33% of Customers accepted the offer in the campaign are "0". 14.50% of Customers accepted the offer in the campaign are "1". 3.70% of Customers accepted the offer in the campaign are "2". 1.96% of Customers accepted the offer in the campaign are "3". 0.49% of Customers accepted the offer in the campaign are "4".

8.Analysis on "NumWebPurchases,NumCatalogPurchases,NumStorePurchases,NumDealsPurchases" Variable.

In [49]:
```
df['NumWebPurchases'].unique()
```

Out[49]:
```
array([ 8,  1,  2,  5,  6,  7,  4,  3, 11,  0, 27, 10,  9, 23, 25],
      dtype=int64)
```

In [50]:
```
df['NumCatalogPurchases'].unique()
```

Out[50]:
```
array([10,  1,  2,  0,  3,  4,  6, 28,  9,  5,  8,  7, 11, 22],
      dtype=int64)
```

In [51]:
```
df['NumStorePurchases'].unique()
```

Out[51]:
```
array([ 4,  2, 10,  6,  7,  0,  3,  8,  5, 12,  9, 13, 11,  1],
      dtype=int64)
```

In [52]:
```
df['NumTotalPurchases'] = df['NumWebPurchases'] + df['NumCatalogPurchases']
df['NumTotalPurchases'].unique()
```

Out[52]:
```
array([25,  6, 21,  8, 19, 22, 10,  2,  4, 16, 15,  5, 26,  9, 13, 12, 43,
       17, 20, 14, 27, 11, 18, 28,  7, 24, 29, 23, 32, 30, 37, 31, 33, 35,
       39,  1, 34,  0, 44], dtype=int64)
```

In [53]:
```
df[['NumTotalPurchases']]
```

Out[53]:

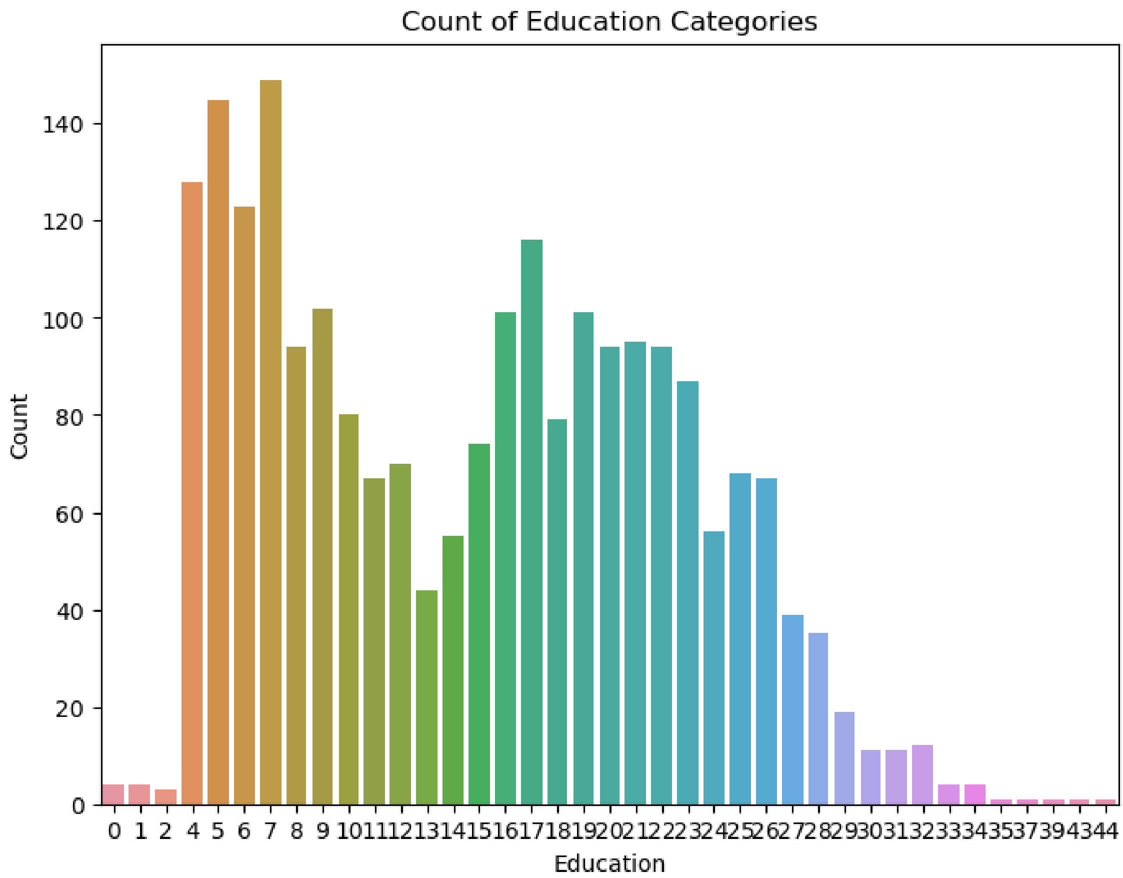
	NumTotalPurchases
0	25
1	6
2	21
3	8
4	19
...	...
2235	18
2236	22
2237	19
2238	23
2239	11

2240 rows × 1 columns

In [54]:
```
df['NumTotalPurchases'].describe()
```

Out[54]:
```
count    2240.000000
mean       14.862054
std         7.677173
min         0.000000
25%         8.000000
50%        15.000000
75%        21.000000
max        44.000000
Name: NumTotalPurchases, dtype: float64
```

In [55]:
```
uni_V('NumTotalPurchases')
```



In [56]:
```
df.head()
```

Out[56]:

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

5 rows × 31 columns

9.Converting the Year_Birth to customer_Age

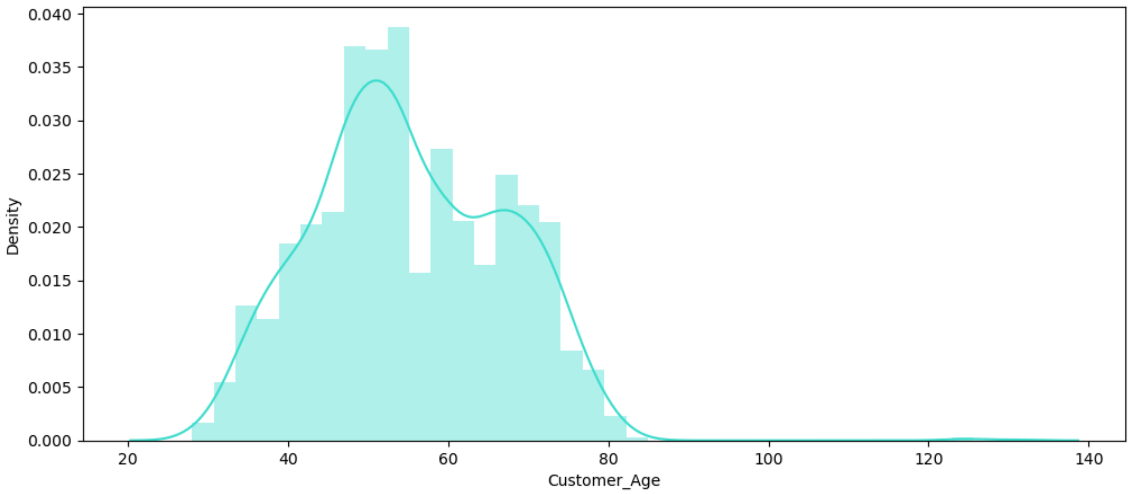
In [57]:
```
#ADDING A COLUMN "customer_Age" IN THE DATAFRAME....
df['Customer_Age'] = (pd.Timestamp('now').year) - df['Year_Birth']
df.head()
```

Out[57]:

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

5 rows × 32 columns

In [58]:
```
plt.figure(figsize=(12,5))
sns.distplot(df["Customer_Age"],color = 'turquoise')
plt.show()
```



Most of the cutomers we have are in middle age i.e between 35-55

In [59]:
```
# Deleting some column to reduce dimension and complexity of model

col_del = ["Year_Birth","ID","AcceptedCmp1" , "AcceptedCmp2", "AcceptedCmp3
df=df.drop(columns=col_del,axis=1)
```

In [60]:
```
df.head()
```

Out[60]:

	Education	Marital_Status	Income	Dt_Customer	Recency	Complain	Response	Kids	E›
0	Post Graduate	Single	58138.0	04-09-2012	58	0	1	0	
1	Post Graduate	Single	46344.0	08-03-2014	38	0	0	2	
2	Post Graduate	Relationship	71613.0	21-08-2013	26	0	0	0	
3	Post Graduate	Relationship	26646.0	10-02-2014	26	0	0	1	
4	Post Graduate	Relationship	58293.0	19-01-2014	94	0	0	1	

In [61]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Education          2240 non-null   object 
 1   Marital_Status     2240 non-null   object 
 2   Income             2240 non-null   float64
 3   Dt_Customer        2240 non-null   object 
 4   Recency            2240 non-null   int64  
 5   Complain           2240 non-null   int64  
 6   Response           2240 non-null   int64  
 7   Kids               2240 non-null   int64  
 8   Expenses           2240 non-null   int64  
 9   TotalAcceptedCmp   2240 non-null   int64  
 10  NumTotalPurchases  2240 non-null   int64  
 11  Customer_Age       2240 non-null   int64  
dtypes: float64(1), int64(8), object(3)
memory usage: 210.1+ KB
```

In the next step, we 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, I am taking this value relative to the most recent customer in the record.

Thus to get the values I must check the newest and oldest recorded dates.

In [62]:
```
df["Dt_Customer"] = pd.to_datetime(df["Dt_Customer"])
dates = []
for i in df["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 [63]:
```
#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)
df["Customer_For"] = days
df['Customer_For'] = df['Customer_For'].apply(lambda x:x.days)
```

In [64]:
```
df.head()
```

Out[64]:

	Education	Marital_Status	Income	Dt_Customer	Recency	Complain	Response	Kids	E
0	Post Graduate	Single	58138.0	2012-04-09	58	0	1	0	
1	Post Graduate	Single	46344.0	2014-08-03	38	0	0	2	
2	Post Graduate	Relationship	71613.0	2013-08-21	26	0	0	0	
3	Post Graduate	Relationship	26646.0	2014-10-02	26	0	0	1	
4	Post Graduate	Relationship	58293.0	2014-01-19	94	0	0	1	

In [65]:
```
df['Customer_For'].describe()
```

Out[65]:
```
count    2240.000000
mean      512.043304
std       232.229893
min         0.000000
25%       340.750000
50%       513.000000
75%       685.250000
max      1063.000000
Name: Customer_For, dtype: float64
```

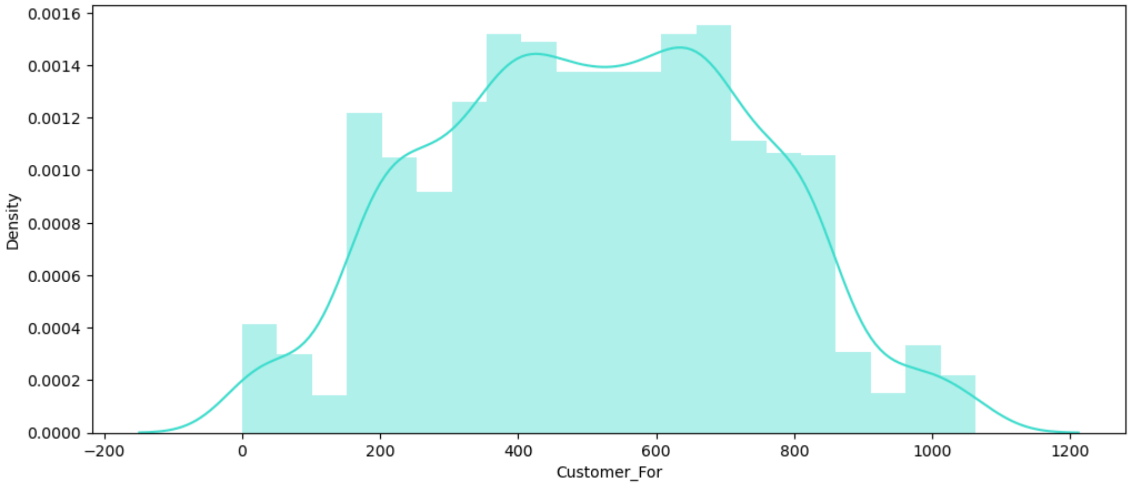
In [66]:
```
df.drop(['Dt_Customer','Recency','Complain','Response'],axis=1,inplace=True
```

In [67]:
```
df.head()
```

Out[67]:

	Education	Marital_Status	Income	Kids	Expenses	TotalAcceptedCmp	NumTotalPurchase
0	Post Graduate	Single	58138.0	0	1617	0	2
1	Post Graduate	Single	46344.0	2	27	0	
2	Post Graduate	Relationship	71613.0	0	776	0	2
3	Post Graduate	Relationship	26646.0	1	53	0	
4	Post Graduate	Relationship	58293.0	1	422	0	1

In [68]:
```
plt.figure(figsize=(12,5))

sns.distplot(df["Customer_For"],color = 'turquoise')
plt.show()
```



Most of the customers are regular to the campaign for 200-850 days

In [69]:
```
df.head()
```

Out[69]:

	Education	Marital_Status	Income	Kids	Expenses	TotalAcceptedCmp	NumTotalPurchase
0	Post Graduate	Single	58138.0	0	1617	0	2
1	Post Graduate	Single	46344.0	2	27	0	
2	Post Graduate	Relationship	71613.0	0	776	0	2
3	Post Graduate	Relationship	26646.0	1	53	0	
4	Post Graduate	Relationship	58293.0	1	422	0	1

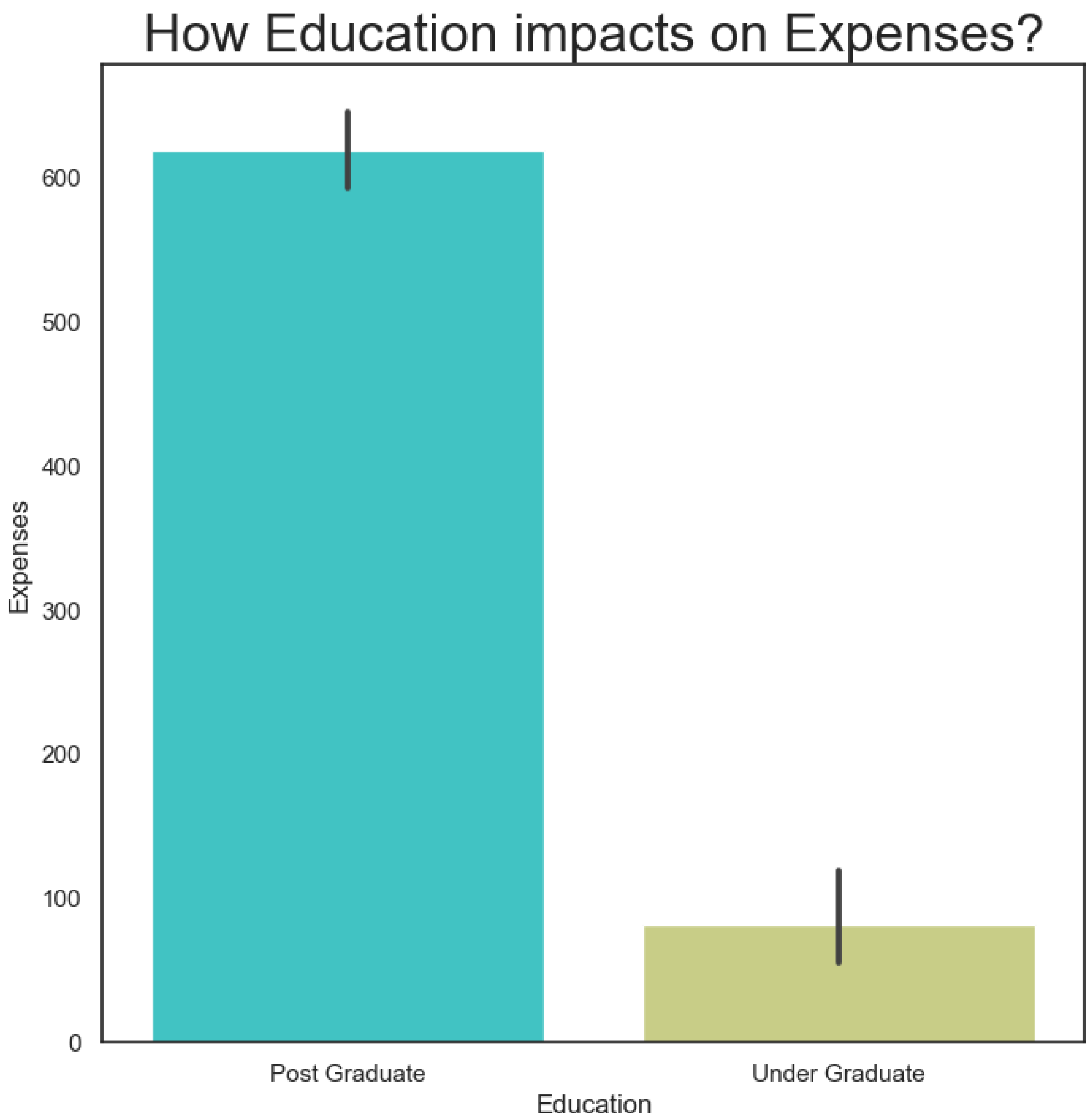
In [70]:
```
df.shape
```

Out[70]:
```
(2240, 9)
```

Bivariate Analysis :-

1.Education vs Expenses

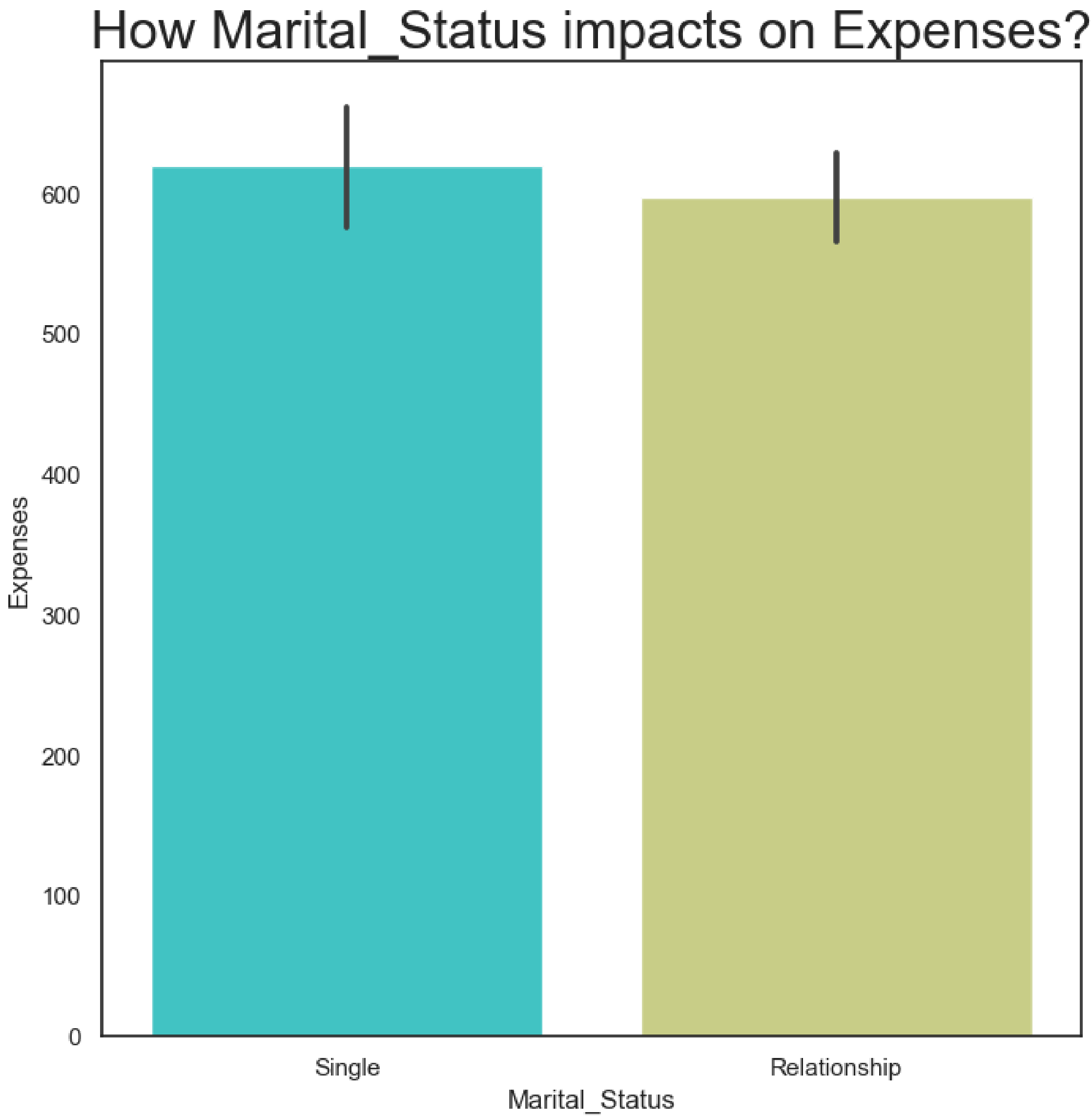
In [71]:
```
sns.set_theme(style="white")
plt.figure(figsize=(8,8))
plt.title("How Education impacts on Expenses?",fontsize=24)
ax = sns.barplot(x="Education", y="Expenses", data=df,palette="rainbow")
```



We observe that the post graduated people spends more than the UG people

2.Marital status vs Expenses

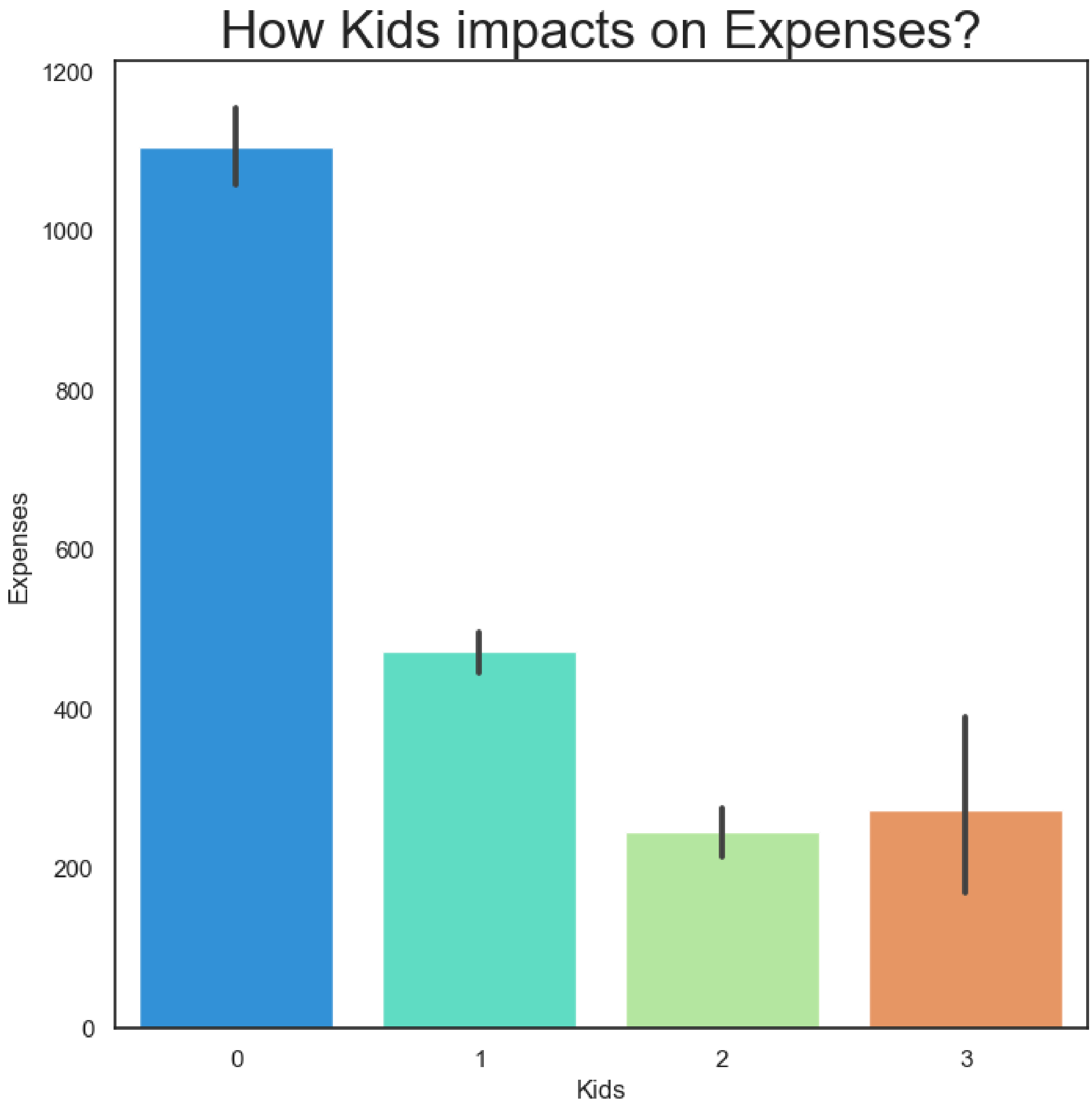
In [72]:
```
sns.set_theme(style="white")
plt.figure(figsize=(8,8))
plt.title("How Marital_Status impacts on Expenses?",fontsize=24)
ax = sns.barplot(x="Marital_Status", y="Expenses", data=df,palette="rainbow
```



We observe that single and married people have the same spendings

3.Kids vs Expenses

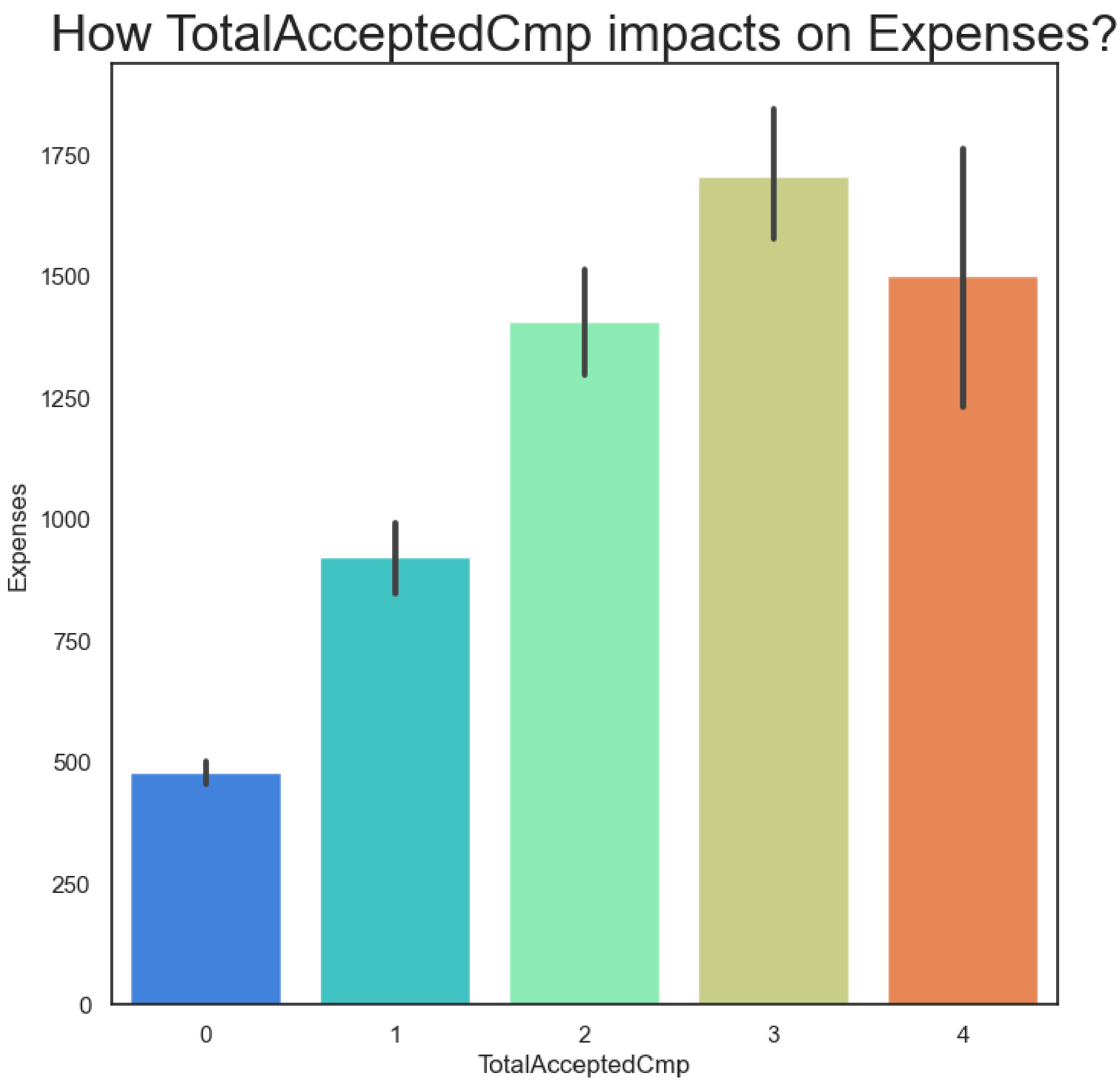
In [73]:
```
sns.set_theme(style="white")
plt.figure(figsize=(8,8))
plt.title("How Kids impacts on Expenses?",fontsize=24)
ax = sns.barplot(x="Kids", y="Expenses", data=df,palette="rainbow")
```



Here we observe some thing different that parents with 1 kid spends more than the parents who are having 2 or 3 kids

4.TotalAcceptedCmp vs Expenses

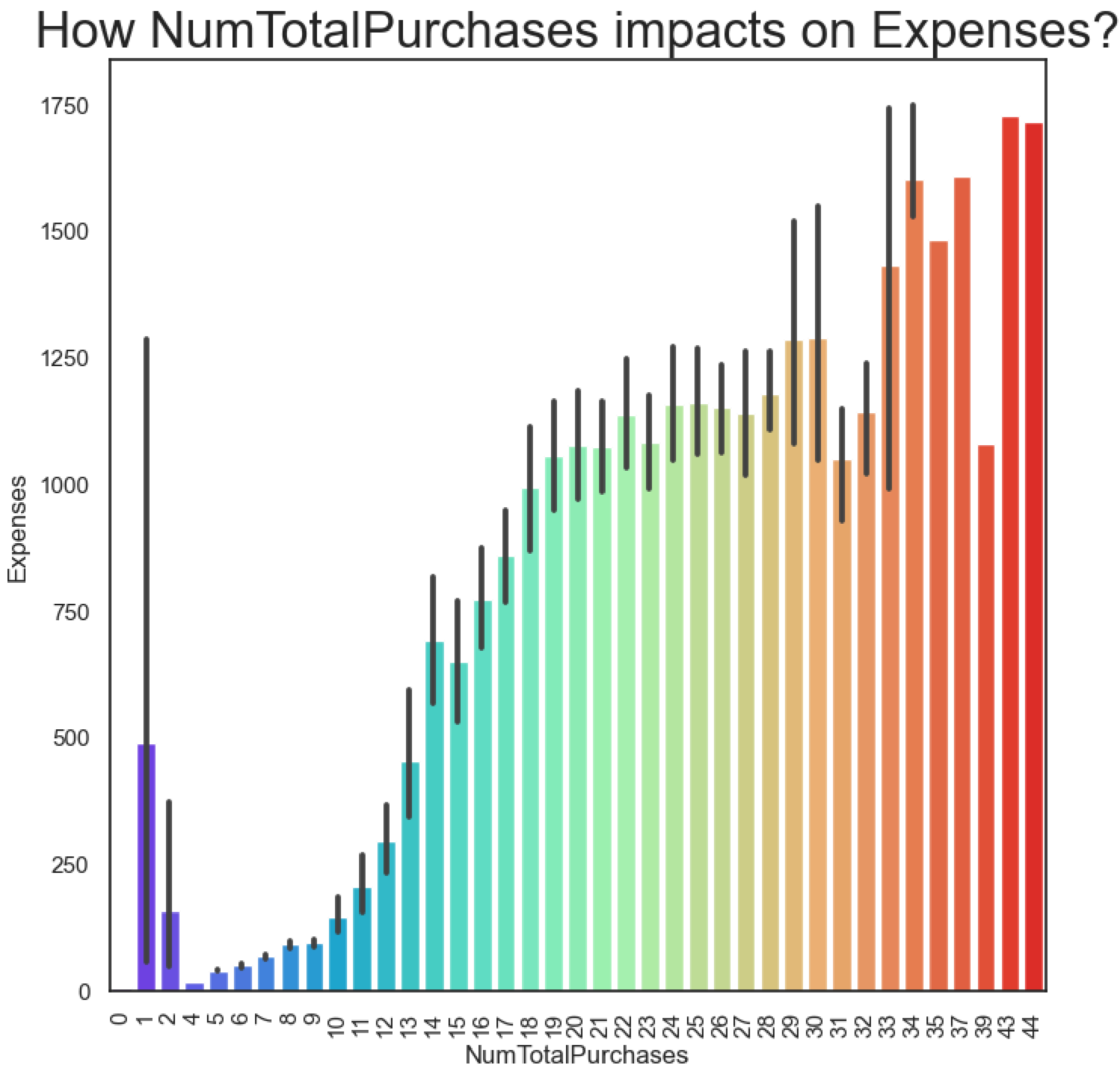
In [74]:
```
sns.set_theme(style="white")
plt.figure(figsize=(8,8))
plt.title("How TotalAcceptedCmp impacts on Expenses?",fontsize=24)
ax = sns.barplot(x="TotalAcceptedCmp", y="Expenses", data=df,palette="rainb
```



hose who accepeted more campaign have more expenses

5.NumTotalPurchases vs Expenses

In [76]:
```
sns.set_theme(style="white")
plt.figure(figsize=(8,8))
plt.title("How NumTotalPurchases impacts on Expenses?",fontsize=24)
plt.xticks(rotation=90)
ax = sns.barplot(x="NumTotalPurchases", y="Expenses", data=df,palette="rain
```



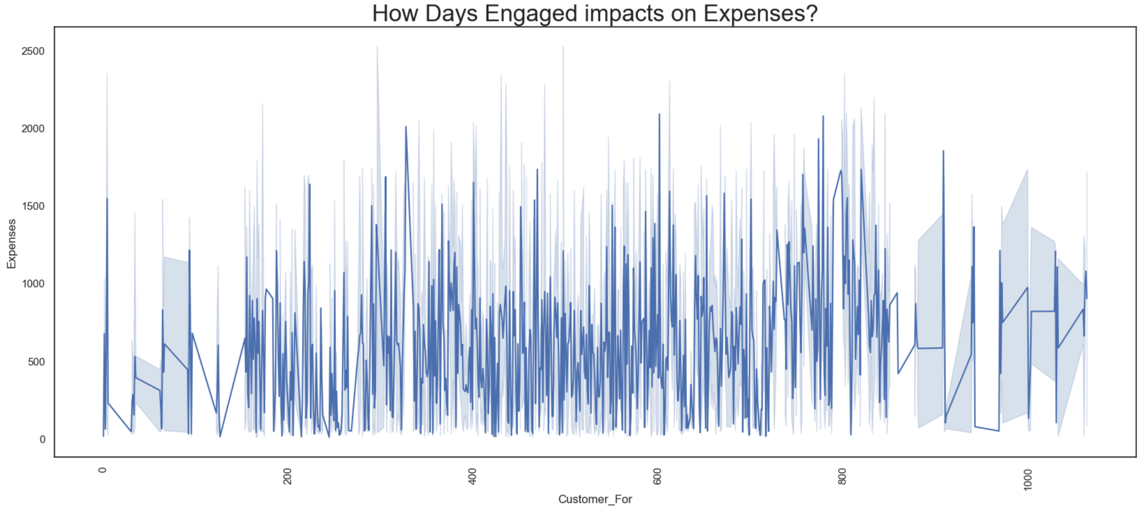
Those who have more purchases have more expenses

6.Day engaged vs Expenses

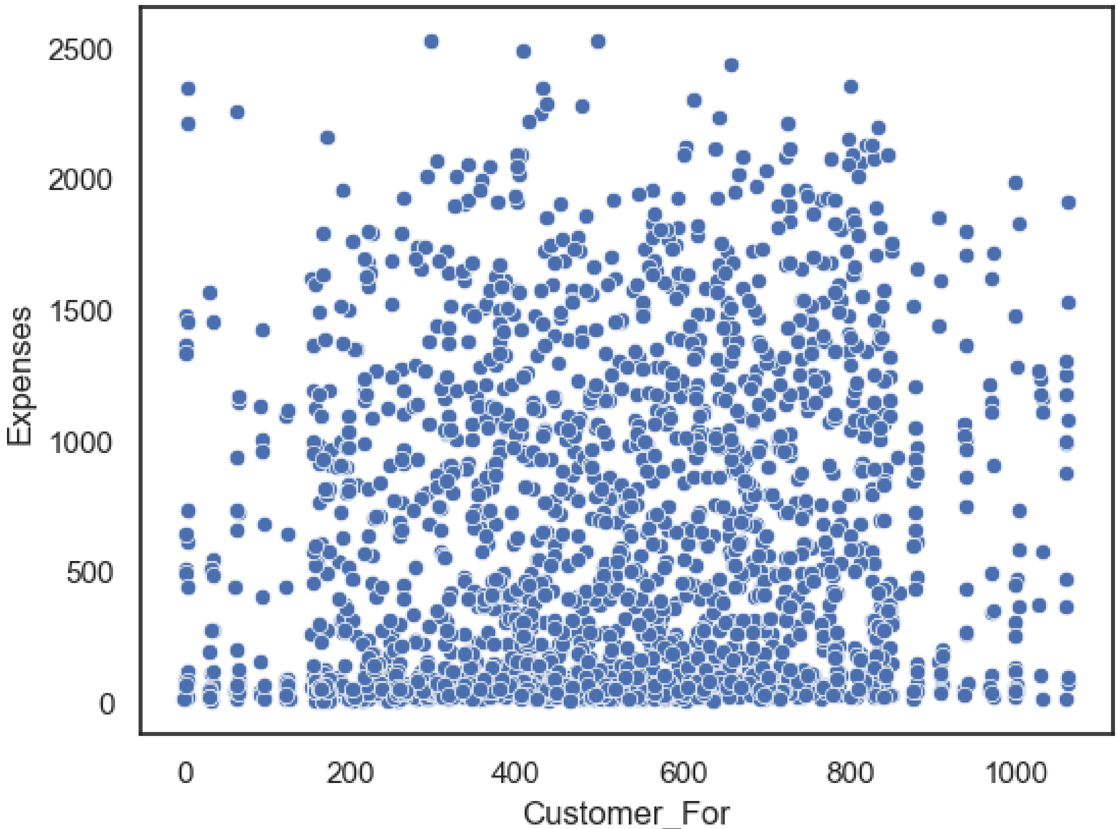
In [77]:
```
df.columns
```

Out[77]:
```
Index(['Education', 'Marital_Status', 'Income', 'Kids', 'Expenses',
       'TotalAcceptedCmp', 'NumTotalPurchases', 'Customer_Age',
       'Customer_For'],
      dtype='object')
```

In [78]:
```
sns.set_theme(style="white")
plt.figure(figsize=(20,8))
plt.title("How Days Engaged impacts on Expenses?",fontsize=24)
plt.xticks(rotation=90)
ax = sns.lineplot(x="Customer_For", y="Expenses", data=df,palette="rainbow"
```



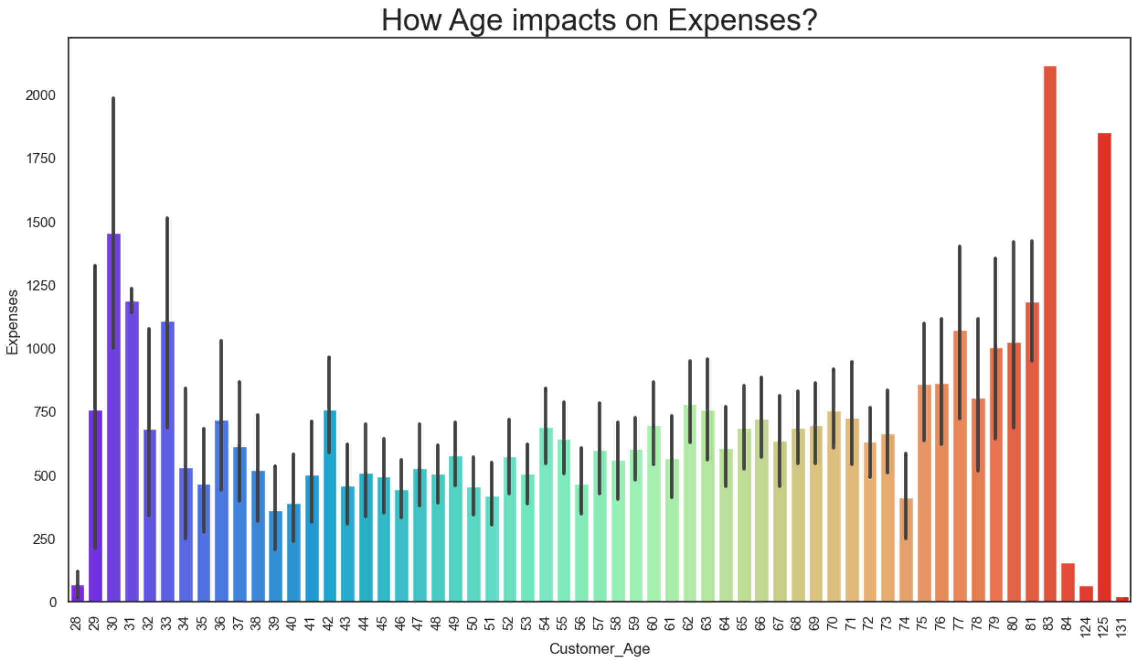
In [80]:
```
sns.scatterplot(x='Customer_For', y='Expenses', data=df)
plt.show()
```



No relationship between days enagaged vs expenses

7.Customer Age vs Expenses

In [81]:
```
sns.set_theme(style="white")
plt.figure(figsize=(15,8))
plt.title("How Age impacts on Expenses?",fontsize=24)
plt.xticks(rotation=90)
ax = sns.barplot(x="Customer_Age", y="Expenses", data=df,palette="rainbow")

plt.show()
```



People who are in middle age have less expenses than others

Remove some outliers present in age and income

In [82]:
```
df['Income'].describe()
```

Out[82]:
```
count      2240.000000
mean      52237.975446
std       25037.955891
min        1730.000000
25%       35538.750000
50%       51381.500000
75%       68289.750000
max      666666.000000
Name: Income, dtype: float64
```

In [83]:
```
df['Customer_For'].describe()
```

Out[83]:
```
count    2240.000000
mean      512.043304
std       232.229893
min         0.000000
25%       340.750000
50%       513.000000
75%       685.250000
max      1063.000000
Name: Customer_For, dtype: float64
```

In [84]:
```
df.shape
```
Out[84]: (2240, 9)

In [85]:
```
df = df[df['Customer_Age'] < 90]
df = df[df['Income'] < 300000]
```

In [86]:
```
df.shape
```
Out[86]: (2236, 9)

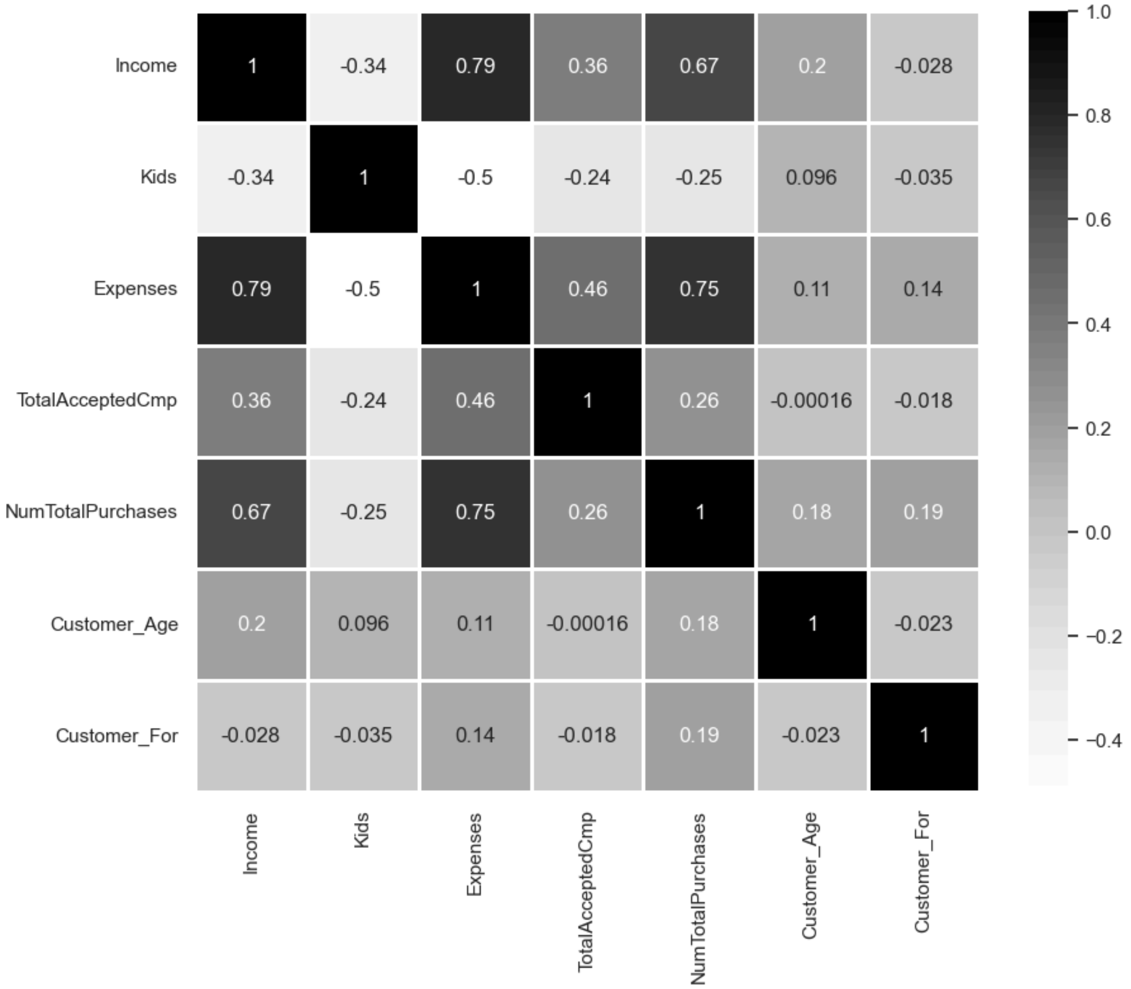
In [87]:
```
df.head()
```
Out[87]:

	Education	Marital_Status	Income	Kids	Expenses	TotalAcceptedCmp	NumTotalPurchase
0	Post Graduate	Single	58138.0	0	1617	0	2
1	Post Graduate	Single	46344.0	2	27	0	
2	Post Graduate	Relationship	71613.0	0	776	0	2
3	Post Graduate	Relationship	26646.0	1	53	0	
4	Post Graduate	Relationship	58293.0	1	422	0	1

Finding the correlation:-

In [88]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), annot=True,cmap = 'Greys',linewidths=1)
```

Out[88]:
```
<Axes: >
```



Income is more positively correlated to Expenses and Number of purchases

Expenses is positively correlated to Income and Number of pur chases and negitively correlated with Kids

In [89]:
```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows
# how to understand word labels.
label_encoder = preprocessing.LabelEncoder()


df['Education'] = label_encoder.fit_transform(df['Education'])
df['Marital_Status'] = label_encoder.fit_transform(df['Marital_Status'])
```

In [91]:
```
df.columns
```

Out[91]:
```
Index(['Education', 'Marital_Status', 'Income', 'Kids', 'Expenses',
       'TotalAcceptedCmp', 'NumTotalPurchases', 'Customer_Age',
       'Customer_For'],
      dtype='object')
```

In [92]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
col_scale = ['Income', 'Kids', 'Expenses',
       'TotalAcceptedCmp', 'NumTotalPurchases', 'Customer_Age', 'Customer_F

df[col_scale] = scaler.fit_transform(df[col_scale])
```

In [93]:
```
df.head()
```

Out[93]:

	Education	Marital_Status	Income	Kids	Expenses	TotalAcceptedCmp	NumTotalPu
0	0	1	0.288947	-1.264308	1.680176	-0.438933	'
1	0	1	-0.262003	1.395139	-0.962202	-0.438933	-'
2	0	0	0.918423	-1.264308	0.282541	-0.438933	(
3	0	0	-1.182183	0.065416	-0.918994	-0.438933	-(
4	0	0	0.296187	0.065416	-0.305762	-0.438933	(

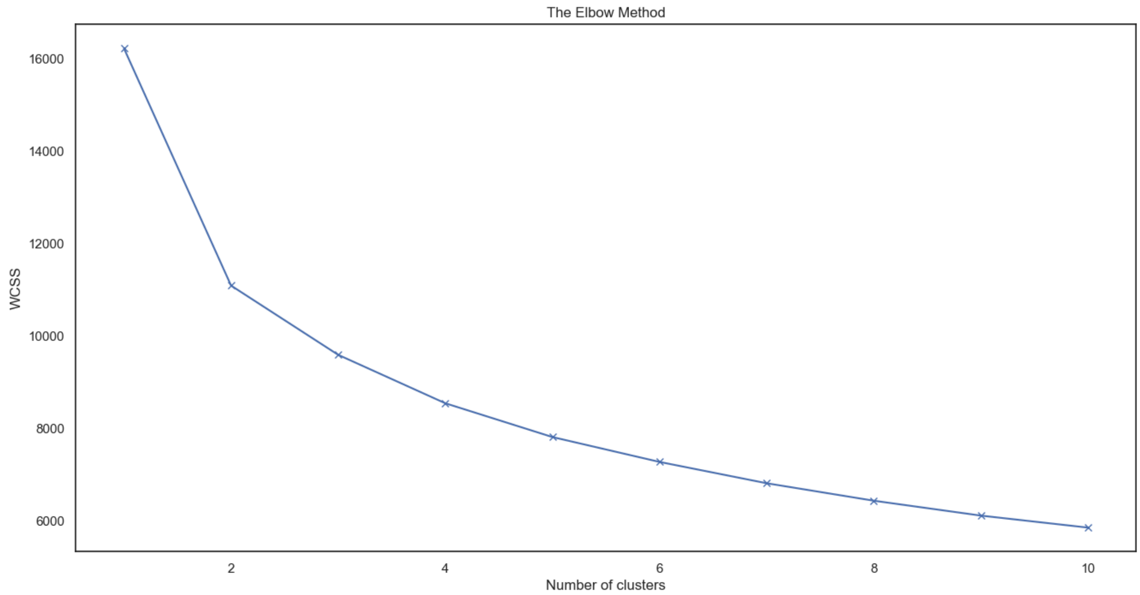
Model Building

K-Means

In [94]:
```
X_0 = df.copy()
```

In [95]:
```
from sklearn.cluster import KMeans
```

In [96]:
```
wcss=[]
for i in range (1,11):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)
    kmeans.fit(X_0)
    wcss.append(kmeans.inertia_)
plt.figure(figsize=(16,8))
plt.plot(range(1,11),wcss, 'bx-')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



We can understand from the plot that cluster = 2 is the best...

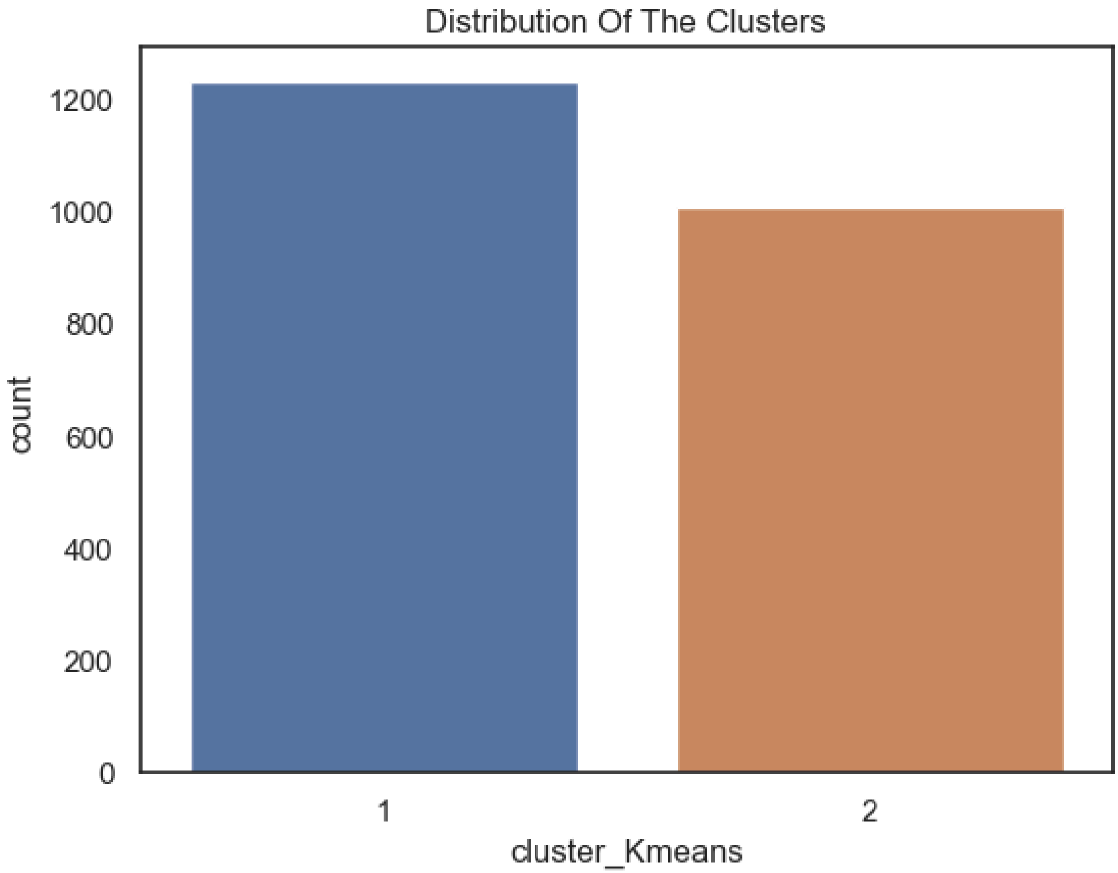
In [97]:
```
# Training a predicting using K-Means Algorithm.

kmeans=KMeans(n_clusters=2, random_state=42).fit(X_0)
pred=kmeans.predict(X_0)


# Appending those cluster value into main dataframe (without standard-scala

X_0['cluster_Kmeans'] = pred + 1
```

In [98]:
```
X_0.head()
```

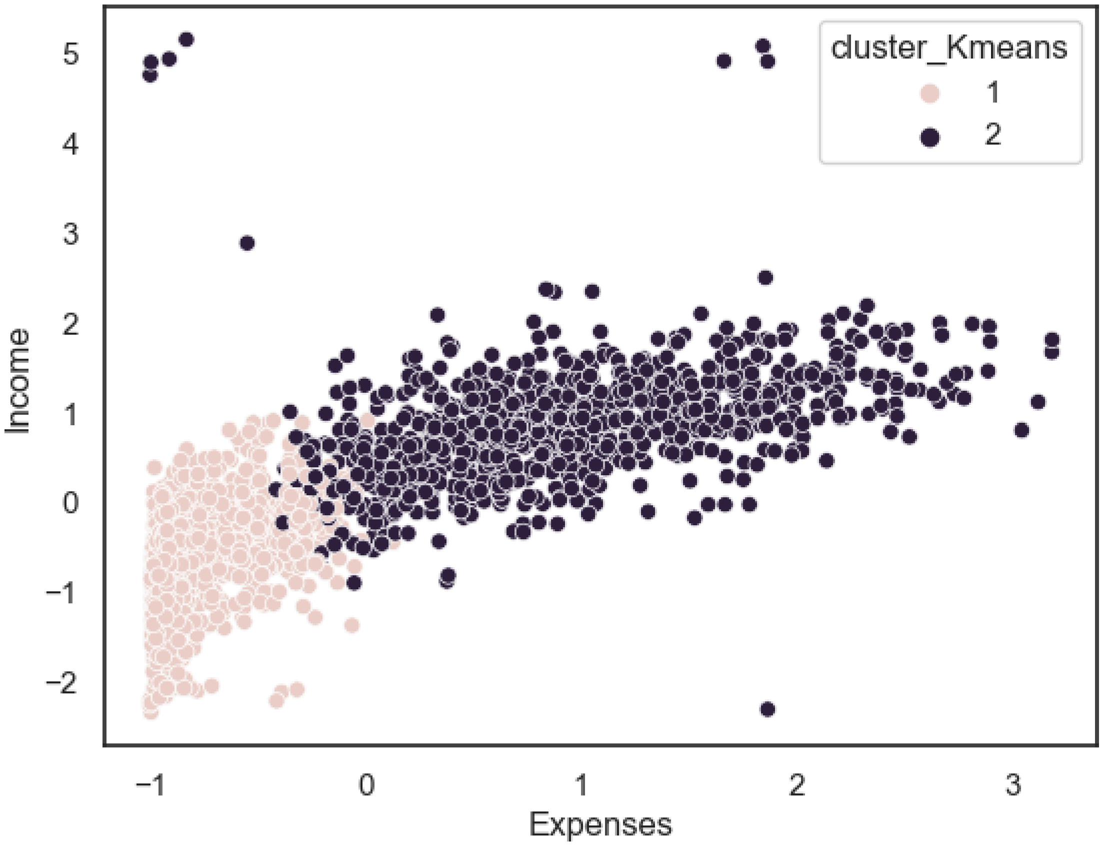
Out[98]:

	Education	Marital_Status	Income	Kids	Expenses	TotalAcceptedCmp	NumTotalPu
0	0	1	0.288947	-1.264308	1.680176	-0.438933	
1	0	1	-0.262003	1.395139	-0.962202	-0.438933	-
2	0	0	0.918423	-1.264308	0.282541	-0.438933	(
3	0	0	-1.182183	0.065416	-0.918994	-0.438933	-(
4	0	0	0.296187	0.065416	-0.305762	-0.438933	(

In [99]:
```
sns.countplot(x=X_0["cluster_Kmeans"])
plt.title("Distribution Of The Clusters")
plt.show()
```



In [101]:
```
sns.scatterplot(x=X_0['Expenses'], y=X_0['Income'], hue=X_0['cluster_Kmeans
plt.show()
```



pca with Agglomerative clustering

In [102]:
```
df.head()
```

Out[102]:

	Education	Marital_Status	Income	Kids	Expenses	TotalAcceptedCmp	NumTotalPu
0	0	1	0.288947	-1.264308	1.680176	-0.438933	'
1	0	1	-0.262003	1.395139	-0.962202	-0.438933	-'
2	0	0	0.918423	-1.264308	0.282541	-0.438933	(
3	0	0	-1.182183	0.065416	-0.918994	-0.438933	-(
4	0	0	0.296187	0.065416	-0.305762	-0.438933	(

In [103]:
```
X_1 = df.copy()
```

In [104]:
```
X_1.head()
```

Out[104]:

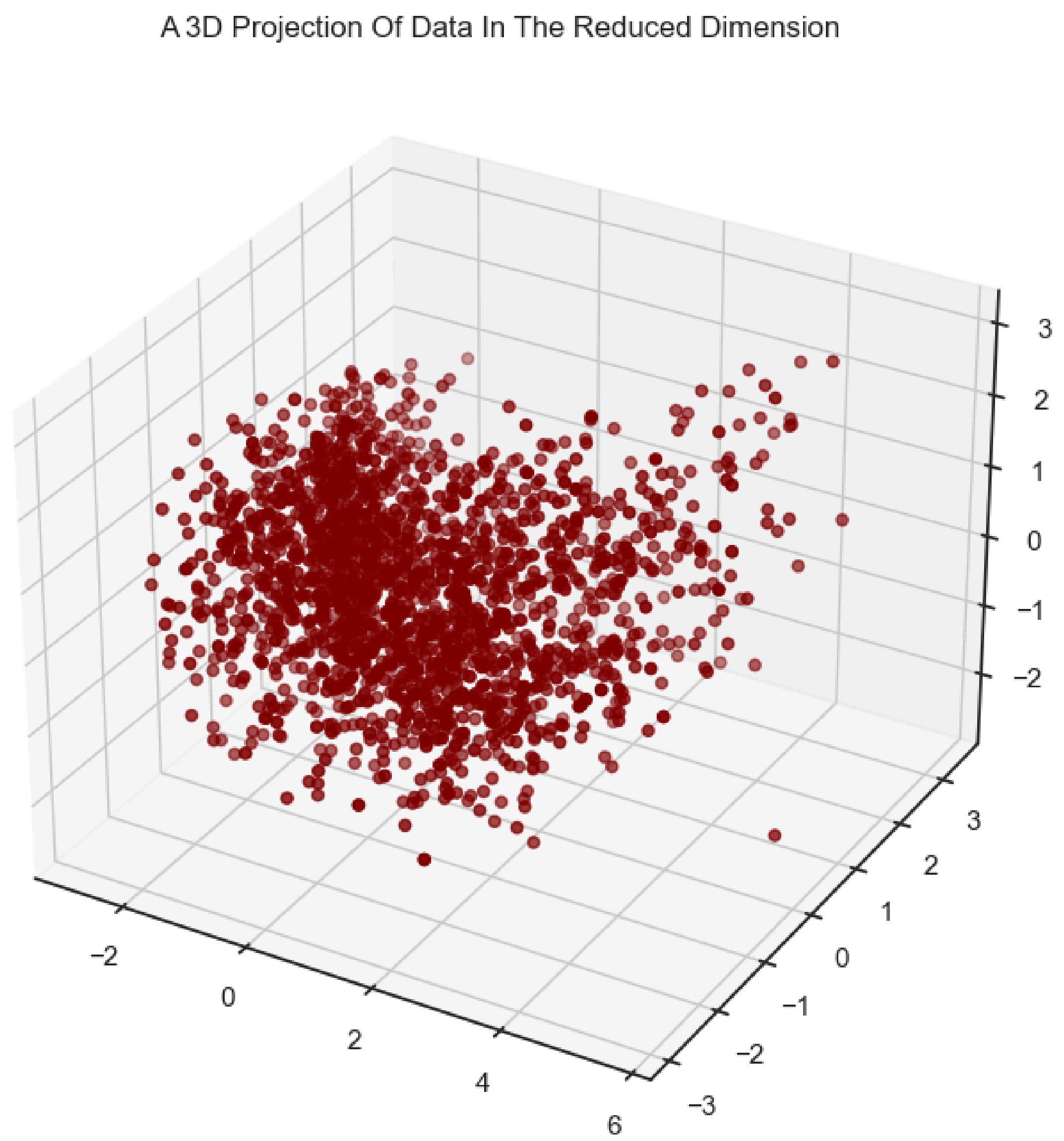
	Education	Marital_Status	Income	Kids	Expenses	TotalAcceptedCmp	NumTotalPu
0	0	1	0.288947	-1.264308	1.680176	-0.438933	'
1	0	1	-0.262003	1.395139	-0.962202	-0.438933	-'
2	0	0	0.918423	-1.264308	0.282541	-0.438933	(
3	0	0	-1.182183	0.065416	-0.918994	-0.438933	-(
4	0	0	0.296187	0.065416	-0.305762	-0.438933	(

In [105]:
```
from sklearn.decomposition import PCA
#Initiating PCA to reduce dimentions aka features to 3
pca = PCA(n_components=3)
pca.fit(X_1)
PCA_ds = pd.DataFrame(pca.transform(X_1), columns=(["col1","col2", "col3"]))
PCA_ds.describe().T
```

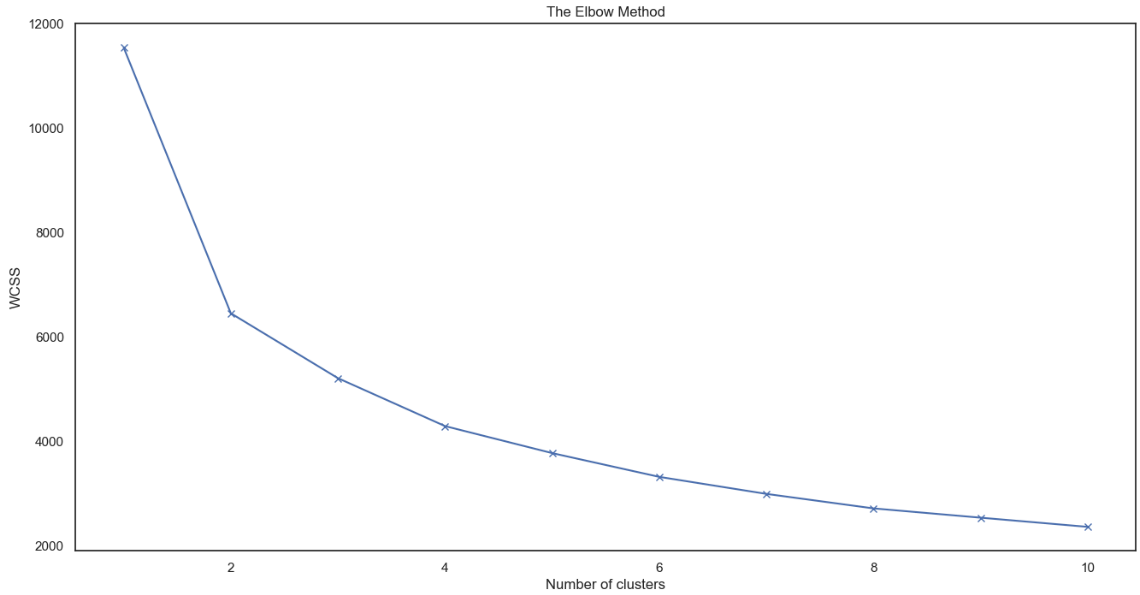
Out[105]:

	count	mean	std	min	25%	50%	75%	max
col1	2236.0	-4.766610e-18	1.726866	-2.826189	-1.609200	-0.271412	1.388004	5.664182
col2	2236.0	-4.448836e-17	1.062690	-2.912907	-0.803814	-0.008308	0.749572	3.380579
col3	2236.0	-2.224418e-17	1.027114	-2.621440	-0.772523	-0.024543	0.767535	3.039268

In [106]:
```
#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 [107]:
```
from sklearn.cluster import AgglomerativeClustering
from sklearn.decomposition import PCA

wcss=[]
for i in range (1,11):
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=42)
    kmeans.fit(PCA_ds)
    wcss.append(kmeans.inertia_)
plt.figure(figsize=(16,8))
plt.plot(range(1,11),wcss, 'bx-')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

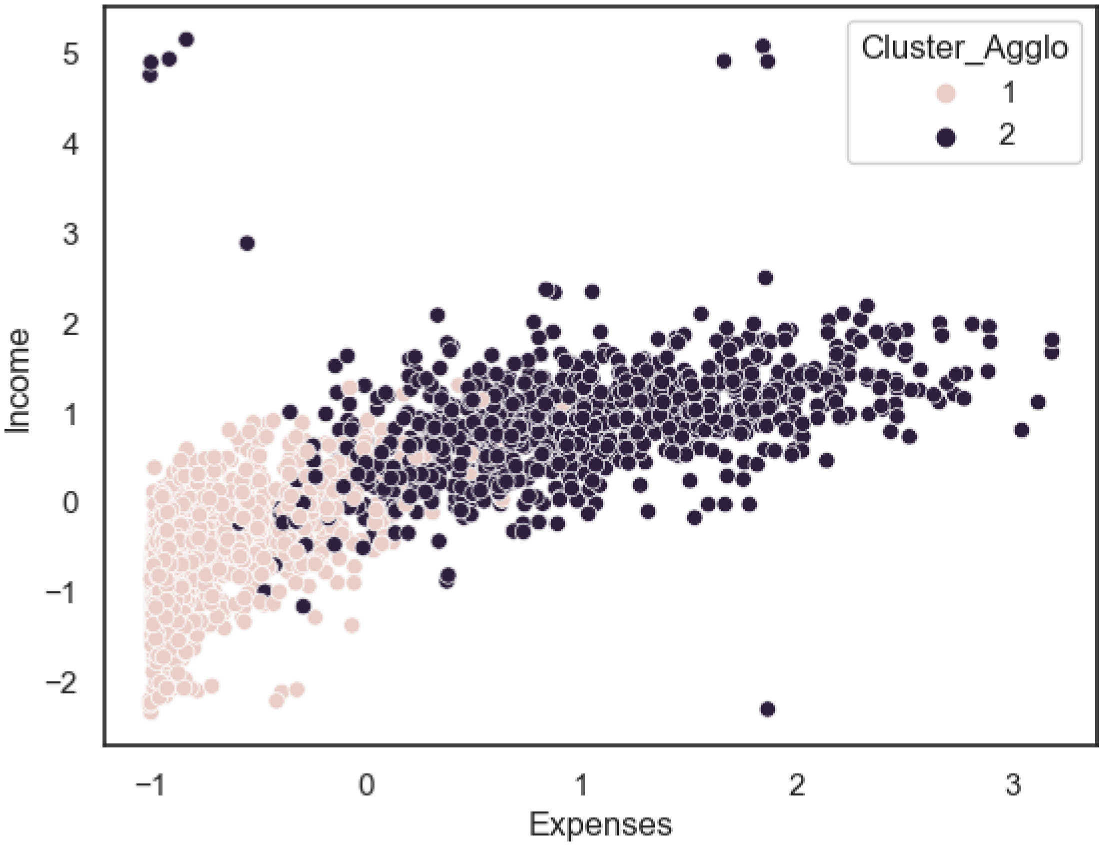


WCSS is the sum of the squared distance between each point and the centroid in a cluster.

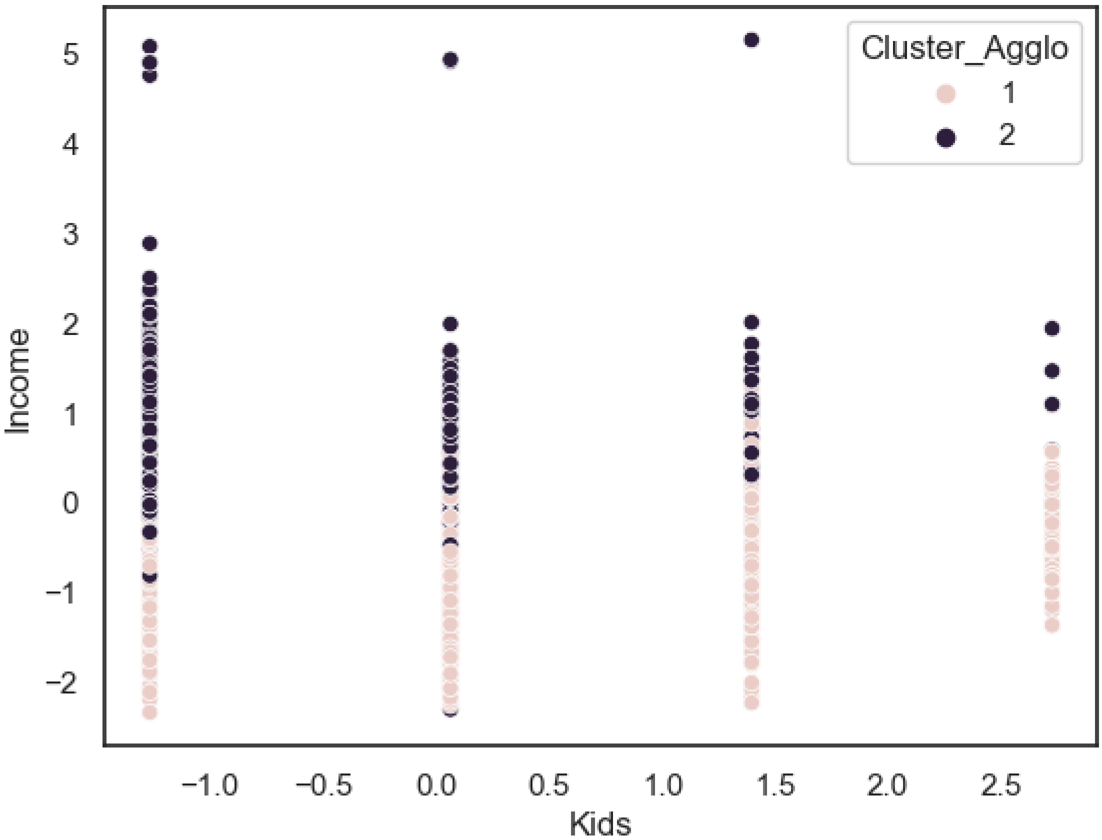
wcss values is more less for k=2 here...so we take k=2

In [108]:
```
#Initiating the Agglomerative Clustering model
AC = AgglomerativeClustering(n_clusters=2)

# 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.
X_1["Cluster_Agglo"]= yhat_AC + 1
```

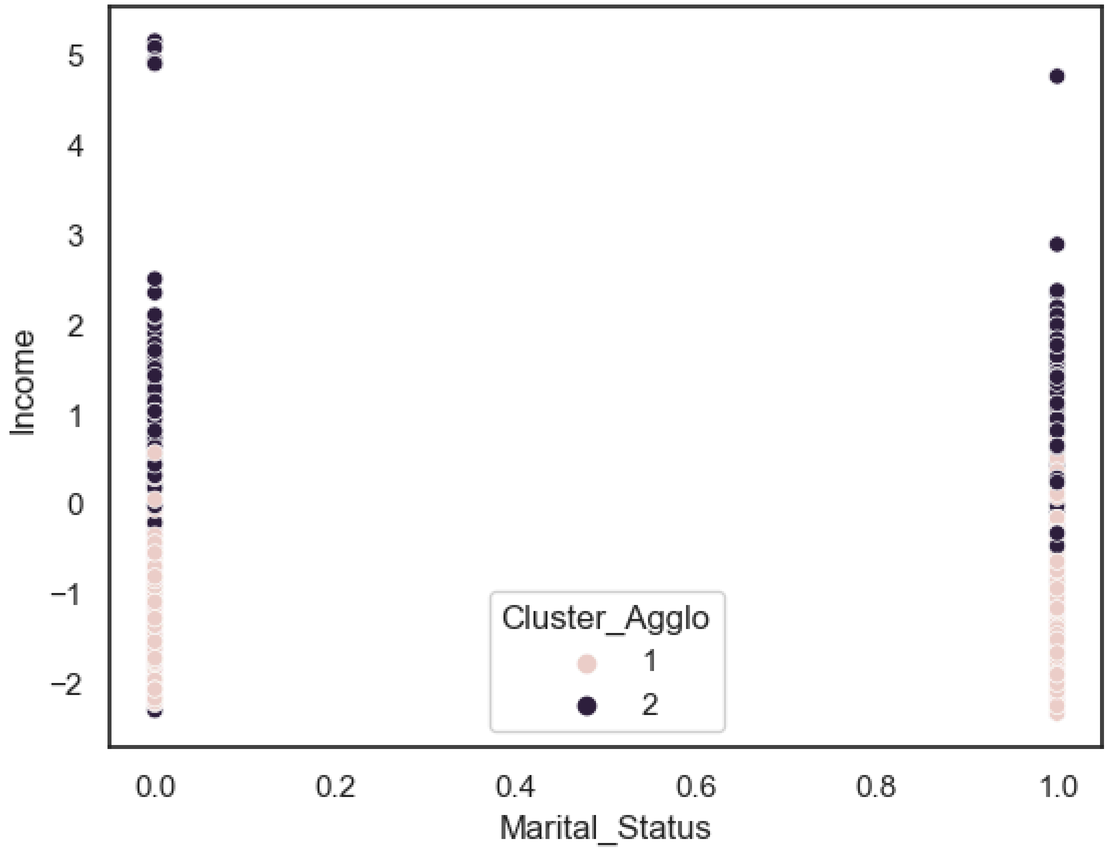
In [110]:
```
sns.scatterplot(x=X_1['Expenses'], y=X_1['Income'], hue=X_1['Cluster_Agglo'
plt.show()
```



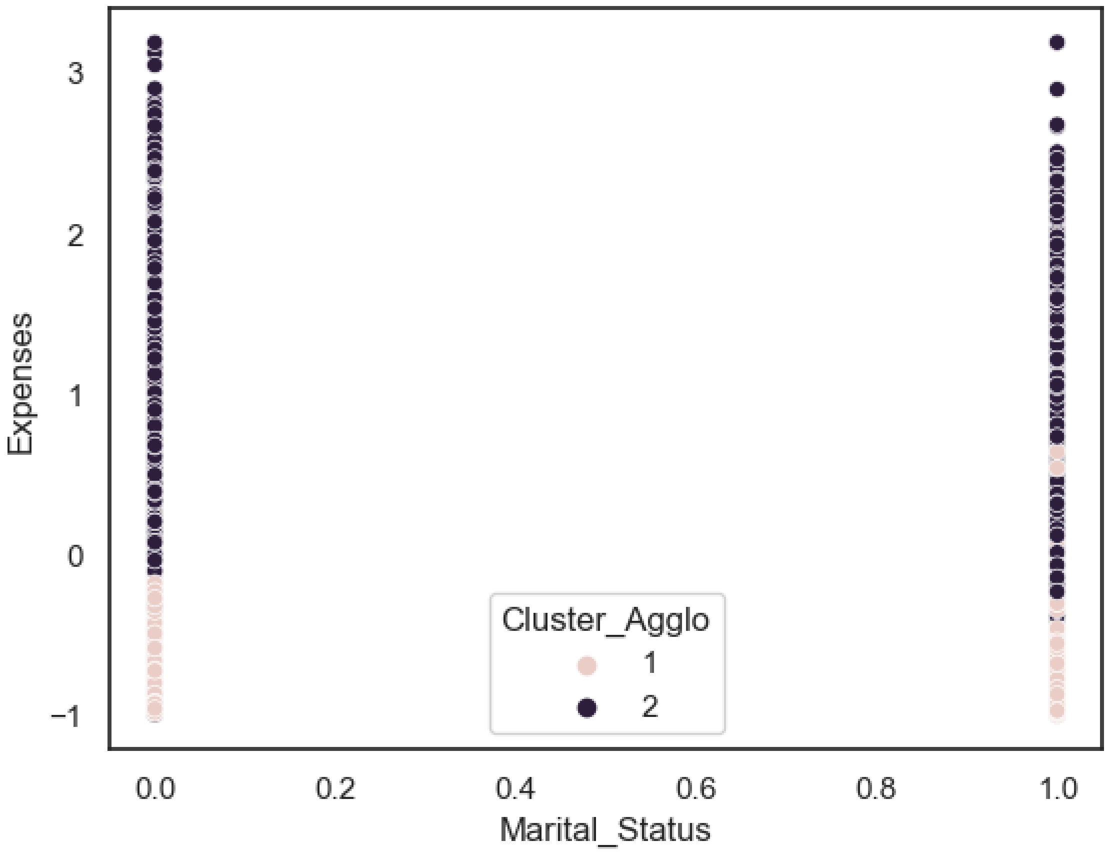
In [111]:
```
sns.scatterplot(x=X_1['Kids'], y=X_1['Income'], hue=X_1['Cluster_Agglo'])
plt.show()
```



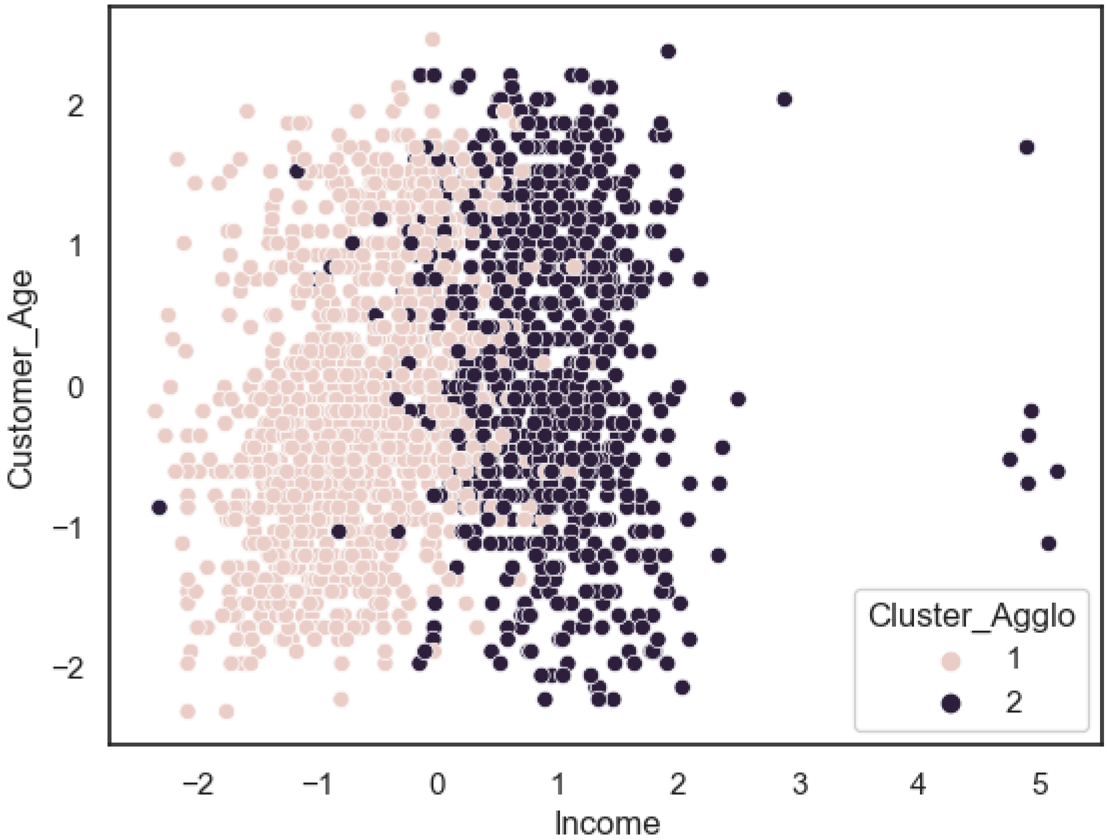
In [112]:
```
sns.scatterplot(x=X_1['Marital_Status'], y=X_1['Income'], hue=X_1['Cluster_
plt.show()
```



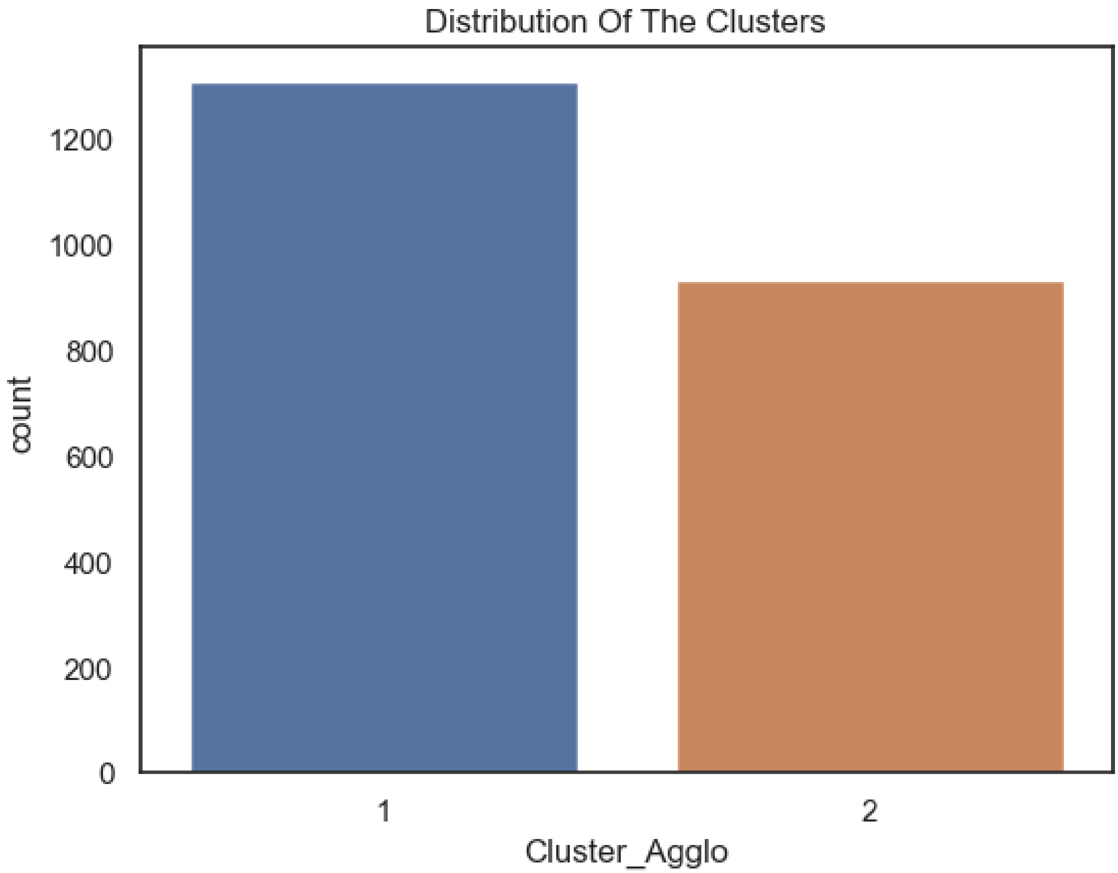
In [113]:
```
sns.scatterplot(x=X_1['Marital_Status'], y=X_1['Expenses'], hue=X_1['Cluste
plt.show()
```



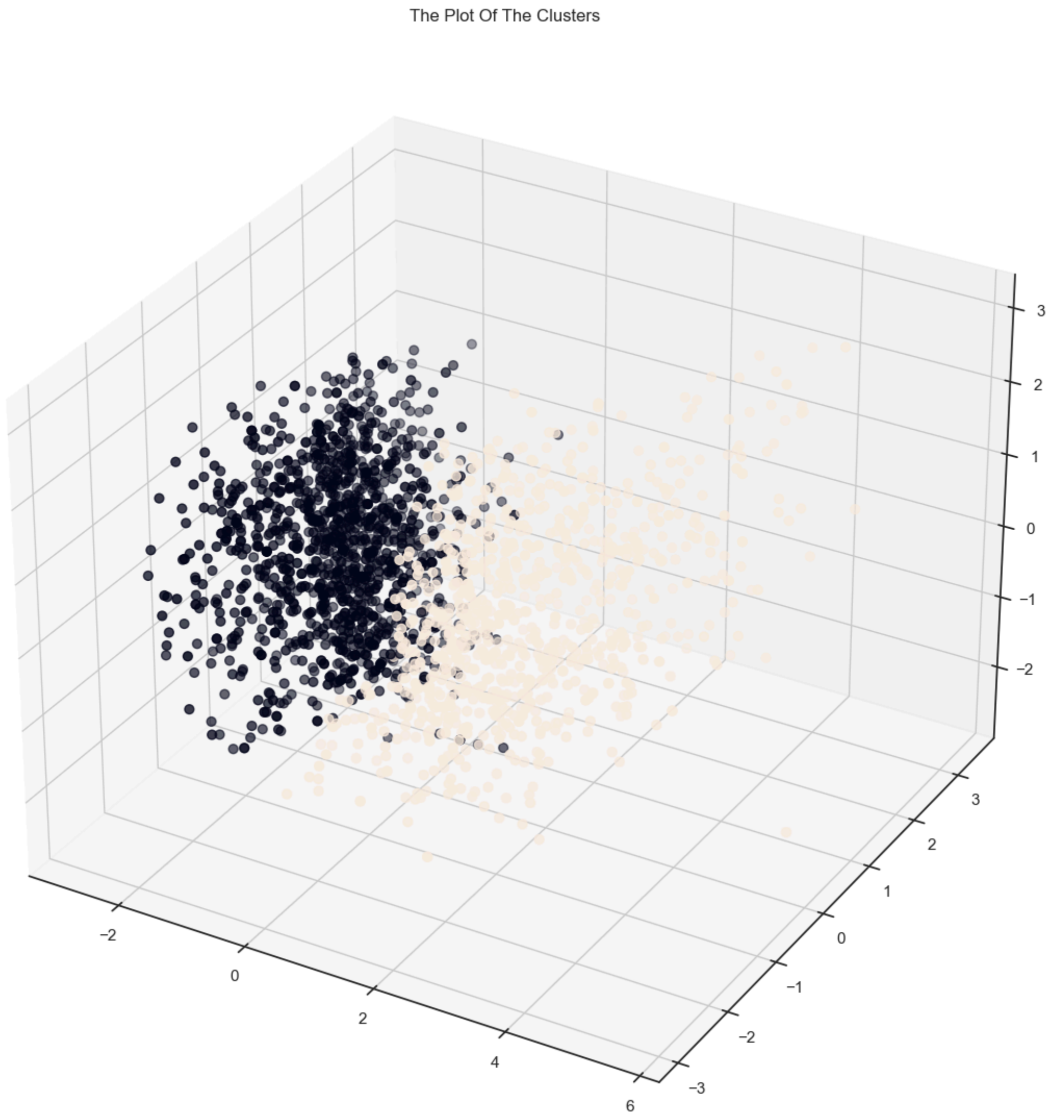
In [114]:
```
sns.scatterplot(x=X_1['Income'], y=X_1['Customer_Age'], hue=X_1['Cluster_Ag
plt.show()
```



In [115]:
```
sns.countplot(x=X_1["Cluster_Agglo"])
plt.title("Distribution Of The Clusters")
plt.show()
```



In [116]:
```
#Plotting the clusters
fig = plt.figure(figsize=(16,14))
ax = plt.subplot(111, projection='3d', label="bla")

ax.scatter(x, y, z, s=40, c=PCA_ds["Clusters"], marker='o')
ax.set_title("The Plot Of The Clusters")

plt.show()
```



Conclusions:

Cluster 1:

People with less expenses

people who are married and parents of more than 3 kids

people which low income

Cluster 2:

people with more expenses

people who are single or parents who have less than 3 kids

people with high income

Age is not the criteria but it is observed to some extent that people who are older fall in this group

So, the customers falling in cluster 2 likes to spend more...so the Firm's can target people falling in cluster 2 for the sale of their Products....

In [ ]:

```

```