```
import pandas as pd
import numpy as np
from matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
# Importing dataset for analysis
df = pd.read_csv("C:/Users/ArunBasker/Good_Job.csv")
df.head()
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	...	marital_Together	marital_Widow	education_2n Cycle	education_Ba
0	58138.0	0	0	58	635	88	546	172	88	88	...	0	0	0	0
1	46344.0	1	1	38	11	1	6	2	1	6	...	0	0	0	0
2	71613.0	0	0	26	426	49	127	111	21	42	...	1	0	0	0
3	26646.0	1	0	26	11	4	20	10	3	5	...	1	0	0	0
4	58293.0	1	0	94	173	43	118	46	27	15	...	0	0	0	0

5 rows × 28 columns

```
df.shape
```

```
# Removing duplicates entries
df.drop_duplicates()
df.head()
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	...	marital_Together	marital_Widow	education_2n Cycle	education_Ba
0	58138.0	0	0	58	635	88	546	172	88	88	...	0	0	0	0
1	46344.0	1	1	38	11	1	6	2	1	6	...	0	0	0	0
2	71613.0	0	0	26	426	49	127	111	21	42	...	1	0	0	0
3	26646.0	1	0	26	11	4	20	10	3	5	...	1	0	0	0
4	58293.0	1	0	94	173	43	118	46	27	15	...	0	0	0	0

5 rows × 28 columns

```
# Attaching null values
df.isnull().sum()
```

```
Income               0
Kidhome              0
Teenhome             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
Response             0
Z_CostContact        0
Z_Revenue            0
Age                  0
marital_Together     0
marital_Divorced     0
marital_Widow        0
education_2n Cycle   0
marital_Together     0
education_Basic      0
education_2n Cycle   0
education_Basic      0
education_Master     0
education_PhD        0
Marital_Status       0
MntRegularProds      0
AcceptedCmpOverall   0
dtype: int64
```

```
# Attaching unique entries
df.nunique()
```

```
Income               163
Kidhome                3
Teenhome               3
Recency               100
MntWines              775
MntFruits             153
MntMeatProducts       558
MntFishProducts       182
MntSweetProducts      177
MntGoldProds          213
NumDealsPurchases      15
NumWebPurchases        15
NumCatalogPurchases    13
NumStorePurchases      14
NumWebVisitsMonth      16
AcceptedCmp3            2
AcceptedCmp4            2
AcceptedCmp5            2
AcceptedCmp1            2
AcceptedCmp2            2
Complain                2
Z_CostContact           1
Z_Revenue               1
Age                    59
marital_Together        2
marital_Divorced        2
marital_Widow           2
education_2n Cycle      2
marital_Together        2
education_Basic         2
education_2n Cycle      2
education_Basic         2
education_Master        2
education_PhD           2
Marital_Status          8
MntRegularProds       776
AcceptedCmpOverall      5
dtype: int64
```

```
# As z_CostContact and Z_Revenue these columns have only a single unique entry and may be not so useful, Deleted these columns
df.drop(['Z_CostContact','Z_Revenue'],
```

```
# Descriptive stats of dataset
df.describe()
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	...	marital_Together	marital_Widow	education_2n Cycle	education_Ba
count	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	...	2240.000000	2240.000000	2240.000000	2240.000000
mean	51622.094643	0.444177	0.506679	49.109679	305.140625	26.402679	166.313616	37.756463	27.156345	44.067143	...	0.257589	0.154467		
std	20713.063826	0.537410	0.544580	28.962111	337.455839	39.784484	217.784507	54.634635	41.139468	51.736231	...	0.437402	0.361442		
min	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000		
25%	35196.500000	0.000000	0.000000	24.000000	24.000000	2.000000	16.000000	3.000000	1.000000	9.000000	...	0.000000	0.000000		
50%	51381.500000	0.000000	0.000000	49.000000	174.000000	8.000000	67.000000	12.000000	8.000000	25.000000	...	0.000000	0.000000		
75%	68281.500000	1.000000	1.000000	74.000000	507.000000	33.000000	232.000000	50.000000	34.000000	56.000000	...	1.000000	0.000000		
max	162397.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	262.000000	321.000000	...	1.000000	1.000000		

8 rows × 37 columns

```
# Node values
df.node[:dropna()]
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	...	marital_Together	marital_Widow	education_2n Cycle	education_Ba
0	79053.	0.0	0.0	96.0	2.0	0.0	7.0	0.0	0.0	3.	...	0.0	0.0	0.0	0.

5 rows × 37 columns

```
# Attaching a column to show whether is a relation or not
def get_relationship(row):
    if row['Marital_Status'] in ['married']:
        return 1
    elif row['Marital_Status'] in['Together']:
        return 1
    else:
        return 0
df['is_relationship'] = df.apply(get_relationship,axis=1)
df.head()
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	...	marital_Widow	education_2n Cycle	education_Basic	Basic
0	58138.0	0	0	58	635	88	546	172	88	88	...	0	0	0	0
1	46344.0	1	1	38	11	1	6	2	1	6	...	0	0	0	0
2	71613.0	0	0	26	426	49	127	111	21	42	...	0	0	1	0
3	26646.0	1	0	26	11	4	20	10	3	5	...	0	0	1	0
4	58293.0	1	0	94	173	43	118	46	27	15	...	0	0	0	0

5 rows × 38 columns

```
# Attaching a column name
df.rename(columns={'education_2n Cycle':'education_2n Cycle'}, inplace=True)
```

```
# Assigning Education status level 1 to 5 in a different one
def Education_status(row):
    if row['education_2n Cycle' ] == 1:
        return '2'
    elif row['education_Basic' ] == 1:
        return '5'
    elif row['education_Graduation' ] == 1:
        return '3'
    elif row['education_Master' ] == 1:
        return '4'
    elif row['education_PhD' ] == 1:
        return '5'
    else:
        return 'Unknown'
df['Education_status'] = df.apply(Education_status, axis=1)
```

SEGMENTATION

```
# Import Sklearn
from Sklearn.cluster import Kmeans
```

```
# Standardizing the scale of the different columns to be used in segmentation

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
cols_to_scale = ['Income', 'MntTotal', 'is_relationship', 'Education_status']
cols_to_scale
df_scaled = df.copy()
df_scaled[cols_to_scale] = scaler.fit_transform(df[cols_to_scale])
```

```
# Attaching a range for cluster values for elbow Method
from sklearn.cluster import Kmeans
inertia_lst = []
for k in range(1,11):
    kmeans = Kmeans(n_clusters=k, n_init=10).fit(df.iloc[:,
                    :].values)
    inertia_lst.append(inertia)
```

```
# Allow method plot visualization

plt.figure(figsize=(7,5))
plt.plot(range(1,11),inertia_lst)
plt.title("Inertia vs. Number of Clusters")
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Inertia")
plt.show()

# It appears to be the ideal clusters of interest
```

```
# Attaching desired number of clusters into our model and creating the segments
model = Kmeans(n_clusters=4, random_state = 7)
cols_to_scale = df.columns[:-1]
df_scaled['clusters'] = model.predict(df_scaled[cols_to_scale])
```

```
C:\Users\ArunBasker\AppData\Local\Programs\Python\Python36\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to *auto* in 1.4. Se
t the value of `n_init` explicitly to suppress the warning
  super(KMeans, self).fit(X, sample_weight=None)
```

```
# Importing PCA and Decomposition
from sklearn import decomposition
pca = decomposition.PCA(n_components=2)
pca_mat = pca.fit_transform(df_scaled[cols_to_scale])
df_scaled['pca1'] = pca_mat[:,0]
df_scaled['pca2'] = pca_mat[:,1]
```

```
# Visualization of the clusters
plt.figure(figsize=(8,6))
sns.scatplot(x='pca1', y='pca2', data=df_scaled, hue = 'clusters', palette='viridis',)
plt.title("Clusters")
plt.xlabel("Principal Component 1 (pc1)")
plt.ylabel("Principal Component 2 (pc2)")
plt.legend(title='Clusters')
```

```
<matplotlib.legend.Legend at 0x1ec3ad7f10>
```

PROPERTIES OF THE CLUSTERS

```
# Attaching of customer id's clusters
customer_num = df_scaled.groupby('clusters')['MntTotal'].count().reset_index()
```

```
plt.figure(figsize=(5,5))
sns.barplot(x='clusters', y='MntTotal', data=customer_num, palette='viridis',)
plt.title("Clusters")
plt.xlabel("Clusters")
plt.ylabel("customer_num")
plt.show()
```

```
# Attaching clusters without modifying the original values of columns used to form clusters

scaler = StandardScaler()
cols_to_scale = ['Income', 'MntTotal', 'is_relationship', 'Education_status']
df_clustered = df.copy()
model = Kmeans(n_clusters=4, random_state = 7)
model.fit(df_scaled[cols_to_scale])
df_clustered['clusters'] = model.predict(df_scaled[cols_to_scale])
df_clustered.head()
```

```
C:\Users\ArunBasker\AppData\Local\Programs\Python\Python36\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to *auto* in 1.4. Se
t the value of `n_init` explicitly to suppress the warning
  super(KMeans, self).fit(X, sample_weight=None)
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	...	education_Basic	education_Graduation	education_Master	ed
0	58138.0	0	0	58	635	88	546	172	88	88	...	0	1	0	0
1	46344.0	1	1	38	11	1	6	2	1	6	...	0	1	0	0
2	71613.0	0	0	26	426	49	127	111	21	42	...	0	1	0	0
3	26646.0	1	0	26	11	4	20	10	3	5	...	0	1	0	0
4	58293.0	1	0	94	173	43	118	46	27	15	...	0	0	0	0

5 rows × 40 columns

```
# Average income of each cluster
df_clustered.groupby('clusters')['Income'].mean()
```

```
plt.figure(figsize=(6,6))
sns.barplot(x='clusters', y='avg_income', data=df_clustered, palette='viridis',)
plt.title("Cluster Avg_income")
plt.xlabel("Cluster")
plt.ylabel("Avg_income")
plt.show()
```

```
C:\Users\ArunBasker\AppData\Local\Programs\Python\Python36\lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning: Mean of empty slice
  boot_dist.append(f(resample, **func_kwargs))
```



```
# Attaching product consumption per cluster
product_cons = df_clustered.groupby('clusters')[['MntWines', 'MntFruits','MntMeatProducts','MntFishProducts','MntSweetProducts','MntGoldProds','MntTotal']].me
Avg_Consumption =
```

	cluster	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	MntTotal
0	0	22.478490	5.500000	20.000000	0.833333	1.779837	10.263702	58.913264
1	1	647.453077	62.727611	441.396667	80.754004	68.031094	74.348548	1310.551686
2	2	148.345238	30.984009	188.583471	41.417548	28.958974	56.291892	715.180833
3	3	109.787537	7.879875	44.533041	13.046562	8.031201	25.462356	183.238960

```
# Descriptive the data using Melt function
melted_df = pd.melt(Avg_Consumption, id_vars=['cluster'], var_name='Product', value_name='Consumption')
```

```
plt.figure(figsize=(11,15))
sns.barplot(x='cluster', y='Consumption', data=melted_df, errorbar=None, hue='Product')
plt.xlabel("Product Consumption by Cluster")
plt.ylabel("Cluster")
plt.title("Product Consumption")
plt.xticks(rotation=0)
plt.legend(title='Product',loc='upper right')
plt.show()
```



```
# Attaching level of Education
Edu_level_Edus = df_clustered.groupby("clusters")["Education_status"].mean()
Edu_level_Edus
```

```
clusters
0    2
1    3
2    4
3    5
Name: Education_status, dtype: object
```

INSIGHTS AND RECOMMENDATION

Cluster 0 This cluster has the highest number of customers belonging to it, but the mean Income is second lowest. They have also been showed to spend the least on the different Products. More marketing targeted towards this cluster should be done and also they should be offered trade discounts to encourage them to buy more as the cluster has numbers and therefore the opportunity to give the business more sales.

Cluster 1 Despite having the smallest population, the customers from this cluster have the highest purchasing power and also earn the secnd highest income on average. Clients from this have a minimum of Secondary education. With the desire to purchase and fairly high income. The business should focus on retaining the clients in this cluster through discounts, targetted marketing and running promotions.

Cluster 2 Customers within this cluster have the second highest purchasing power however much the average income is the least. The minimum level of Education is also 2. These clients are limited but their income but are a great group to target at the beginnigend of the month to maximise sales.

Cluster 3 This cluster starts the highest income but their desire to purchase is just low. This could be due to failure of the business to sufficiently influence these customers. The business should therefore try to convert these clients into paying customers as they have a huge potential through offers, special discounts etc.