Import Libraries

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
import regex as r
from datetime import datetime, time, date
from dateutil import parser
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
```

In [3]:
```
## Read the csv file by impoting into pandas
```

In [4]:
```
df = pd.read_excel('../streamlit_app_marketing campaign-engine/marketing_campaign.xlsx')
df.shape
```

Out[4]:
```
(2240, 29)
```

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

Out[5]:

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

5 rows × 29 columns

PREPROCESSING

check null values

In [6]:
```
df.isna().sum().sum()
```

Out[6]: 24

drop null values

In [7]:
```
df.dropna(inplace= True)
```

Just checking columns

In [8]:
```
print([i for i in df.columns.values])
```

```
['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome', 'Teenhome', 'Dt_Customer', 'Recenc
y', 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds', 'Num
DealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth', 'Accep
tedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_R
evenue', 'Response']
```

lets sum up all the columns with common features

In [9]:
```
print(dict(df.dtypes))
```

```
{'ID': dtype('int64'), 'Year_Birth': dtype('int64'), 'Education': dtype('O'), 'Marital_Status': dtype('O'),
'Income': dtype('float64'), 'Kidhome': dtype('int64'), 'Teenhome': dtype('int64'), 'Dt_Customer': dtype
('O'), 'Recency': dtype('int64'), 'MntWines': dtype('int64'), 'MntFruits': dtype('int64'), 'MntMeatProduct
s': dtype('int64'), 'MntFishProducts': dtype('int64'), 'MntSweetProducts': dtype('int64'), 'MntGoldProds':
dtype('int64'), 'NumDealsPurchases': dtype('int64'), 'NumWebPurchases': dtype('int64'), 'NumCatalogPurchase
s': dtype('int64'), 'NumStorePurchases': dtype('int64'), 'NumWebVisitsMonth': dtype('int64'), 'AcceptedCmp
3': dtype('int64'), 'AcceptedCmp4': dtype('int64'), 'AcceptedCmp5': dtype('int64'), 'AcceptedCmp1': dtype
('int64'), 'AcceptedCmp2': dtype('int64'), 'Complain': dtype('int64'), 'Z_CostContact': dtype('int64'), 'Z_
Revenue': dtype('int64'), 'Response': dtype('int64')}
```

In [10]:
```
df.columns[df.columns.str.contains('Mnt')]
```

Out[10]:
```
Index(['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',
       'MntSweetProducts', 'MntGoldProds'],
      dtype='object')
```

In [11]:
```
month_purchase = df.loc[:,['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts','MntSweetProducts',
month_purchase['mnt_total']= month_purchase.values.sum(axis =1)
month_purchase.head()
```

Out[11]:

	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGoldProds	mnt_total
0	635	88	546	172	88	88	1617
1	11	1	6	2	1	6	27
2	426	49	127	111	21	42	776
3	11	4	20	10	3	5	53
4	173	43	118	46	27	15	422

In [12]:
```
df.columns[df.columns.str.contains('Num')]
```

Out[12]:
```
Index(['NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases',
       'NumStorePurchases', 'NumWebVisitsMonth'],
      dtype='object')
```

In [13]:
```
num_purchase = df.loc[:,['NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases','NumStorePurchases']]
num_purchase['num_total']= num_purchase.values.sum(axis =1)
num_purchase.head()
```

Out[13]:

	NumDealsPurchases	NumWebPurchases	NumCatalogPurchases	NumStorePurchases	num_total
0	3	8	10	4	25
1	2	1	1	2	6
2	1	8	2	10	21
3	2	2	0	4	8
4	5	5	3	6	19

In [14]:
```
df_n = df.loc[:,['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome', 'Teenhome', 'Dt_Cus
df_n.head()
```

Out[14]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	NumWebVisitsMonth
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	58	7
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	38	5
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	26	4
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	26	6
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	94	5

Concat the Mnt and Purchase data frames

In [15]:
```
df2 = pd.concat([df_n,num_purchase.num_total,month_purchase.mnt_total],axis =1)
```

In [16]:
```
df2.head()
```

Out[16]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	NumWebVisitsMonth	num_tota
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	58	7	25
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	38	5	(
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	26	4	2
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	26	6	8
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	94	5	19

lets get the current age from the data

In [17]:
```
current_year = 2022
df2['age'] = current_year - df2.Year_Birth
df2.head()
```

Out[17]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	NumWebVisitsMonth	num_tota
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	58	7	2
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	38	5	
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	26	4	2
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	26	6	
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	94	5	1

In [18]:
```
df2.Marital_Status.value_counts()
```

Out[18]:
```
Married     857
Together    573
Single      471
Divorced    232
Widow        76
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64
```

lets group marital status into couple and single

In [19]:
```
marital_state = []

for i in df2.Marital_Status:
    if i == 'Married' or i =='Together':
        i = 'couple'
        marital_state.append(i)

    else:
        i ='single'
        marital_state.append(i)
```

In [20]:
```
df2['marital_class'] = marital_state
```

In [21]:
```
df2.head()
```

Out[21]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	NumWebVisitsMonth	num_tota
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	58	7	2
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	38	5	
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	26	4	2
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	26	6	
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	94	5	1

lets add the total number of children for each customer

In [22]:
```
df2['total_kids'] = df2.Kidhome + df2.Teenhome
```

In [23]:
```
df2.total_kids
```

Out[23]:
```
0       0
1       2
2       0
3       1
4       1
       ..
2235    1
2236    3
2237    0
2238    1
2239    2
Name: total_kids, Length: 2216, dtype: int64
```

now lets pust together the required dfs

In [24]:
```
df2.head()
```

Out[24]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	NumWebVisitsMonth	num_tota
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	58	7	2
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	38	5	
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	26	4	2
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	26	6	
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	94	5	1

lets drop the cell we dont need

In [25]:
```
df3= df2.drop(["Year_Birth","Kidhome","Teenhome","Dt_Customer","Marital_Status"], axis =1)
df3.rename({"num_total":"purchase_channel","mnt_total":"total_purchases"},axis =1).head()
```

Out[25]:

	ID	Education	Income	Recency	NumWebVisitsMonth	purchase_channel	total_purchases	age	marital_class	total_kids
0	5524	Graduation	58138.0	58	7	25	1617	65	single	0
1	2174	Graduation	46344.0	38	5	6	27	68	single	2
2	4141	Graduation	71613.0	26	4	21	776	57	couple	0
3	6182	Graduation	26646.0	26	6	8	53	38	couple	1
4	5324	PhD	58293.0	94	5	19	422	41	couple	1

In [26]:
```
df3.columns
```

Out[26]:
```
Index(['ID', 'Education', 'Income', 'Recency', 'NumWebVisitsMonth',
       'num_total', 'mnt_total', 'age', 'marital_class', 'total_kids'],
      dtype='object')
```

lets label encode the non numerical columns

In [27]:
```
le = LabelEncoder()
df3.Education= le.fit_transform(df3.Education)
df3.marital_class = le.fit_transform(df3.marital_class)
df3.head()
```

Out[27]:

	ID	Education	Income	Recency	NumWebVisitsMonth	num_total	mnt_total	age	marital_class	total_kids
0	5524	2	58138.0	58	7	25	1617	65	1	0
1	2174	2	46344.0	38	5	6	27	68	1	2
2	4141	2	71613.0	26	4	21	776	57	0	0
3	6182	2	26646.0	26	6	8	53	38	0	1
4	5324	4	58293.0	94	5	19	422	41	0	1

In [28]:
```
X= df3.drop(['ID','Education','Recency','marital_class','age'],axis =1)
X.head()
```

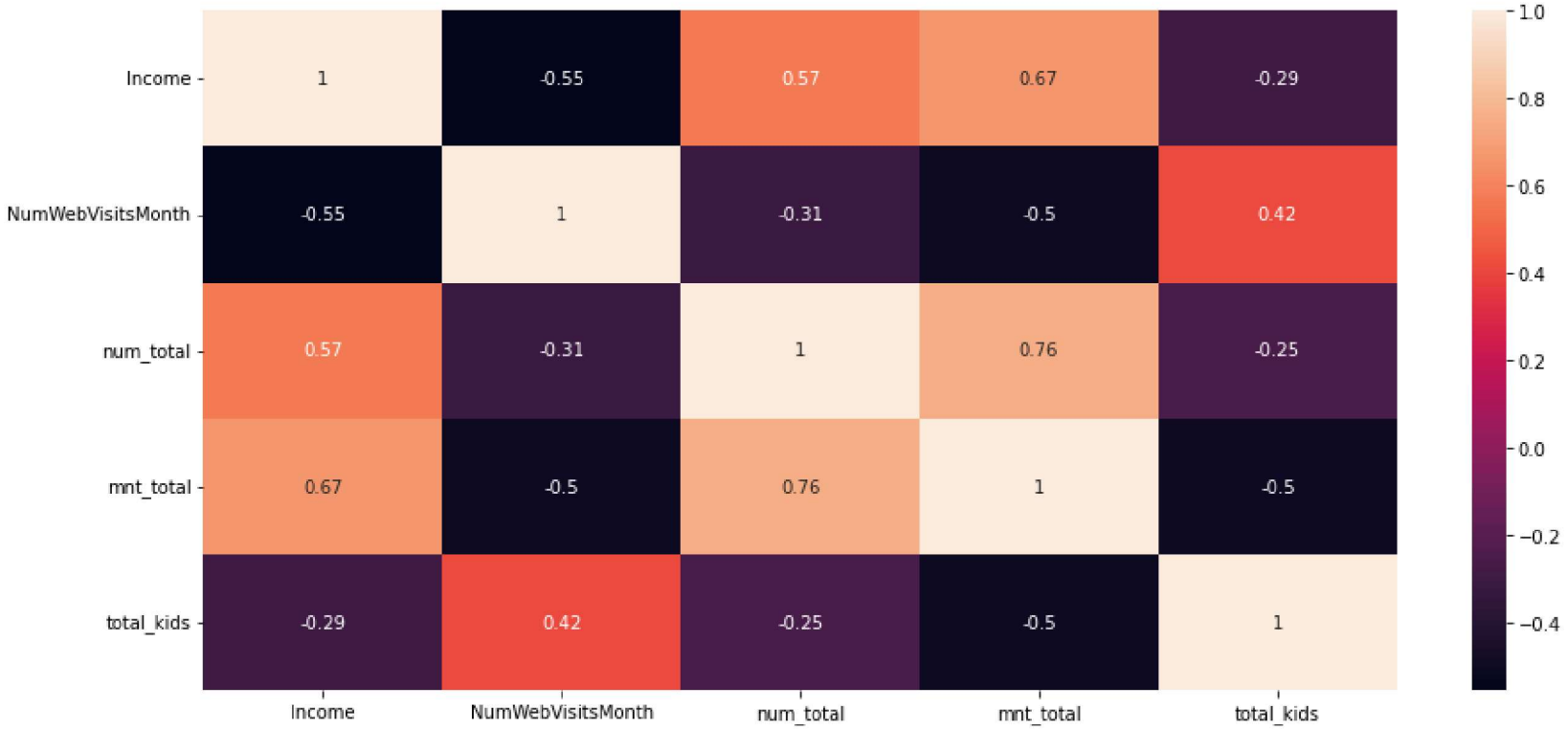
Out[28]:

	Income	NumWebVisitsMonth	num_total	mnt_total	total_kids
0	58138.0	7	25	1617	0
1	46344.0	5	6	27	2
2	71613.0	4	21	776	0
3	26646.0	6	8	53	1
4	58293.0	5	19	422	1

correlation of features

In [29]:
```
plt.subplots(figsize = (15,7))
sns.heatmap(X.corr(),annot = True)
```

Out[29]: <AxesSubplot:>



normalizing with standard sclar

In [30]:
```
mc= MinMaxScaler()
scaledX = mc.fit_transform(X)
scaledX
```

Out[30]:
```
array([[0.08483222, 0.35      , 0.56818182, 0.63968254, 0.        ],
       [0.06709518, 0.25      , 0.13636364, 0.00873016, 0.66666667],
       [0.10509733, 0.2       , 0.47727273, 0.30595238, 0.        ],
       ...,
       [0.08309221, 0.3       , 0.43181818, 0.49047619, 0.        ],
       [0.10153609, 0.15      , 0.52272727, 0.33253968, 0.33333333],
       [0.07690815, 0.35      , 0.25      , 0.06626984, 0.66666667]])
```

feature reduction with PCA

In [31]:
```
# lets reduce the features
pca = PCA(n_components=None, random_state=42)
pcaX= pca.fit_transform(scaledX)
pcaX
```

Out[31]:
```
array([[ 5.26546470e-01,  3.58640445e-02,  1.80453378e-01,
         9.53195421e-02, -1.20990349e-02],
       [-4.37776547e-01,  7.51054634e-02, -1.39229133e-01,
         2.23122976e-02,  6.37150995e-03],
       [ 3.03045503e-01, -1.45329334e-01,  5.41650875e-02,
        -1.11907310e-01,  1.74903757e-02],
       ...,
       [ 3.85784959e-01, -9.29040199e-02,  1.00247411e-01,
         7.35753066e-02, -4.36745840e-03],
       [ 1.48830599e-01,  1.42503837e-01, -4.30425763e-02,
        -1.13631521e-01, -2.38711406e-04],
       [-3.76970293e-01,  1.56696120e-01, -1.35249518e-02,
         2.45280681e-02,  1.89987126e-02]])
```

In [32]:
```
pca.explained_variance_ratio_.cumsum()
```

Out[32]:
```
array([0.64638856, 0.88458113, 0.95084549, 0.99603164, 1.        ])
```

In [33]:
```
pca = PCA(n_components=3)
pcaX = pca.fit_transform(X)
pcaX = pd.DataFrame(pcaX,)
pcaX
```

Out[33]:

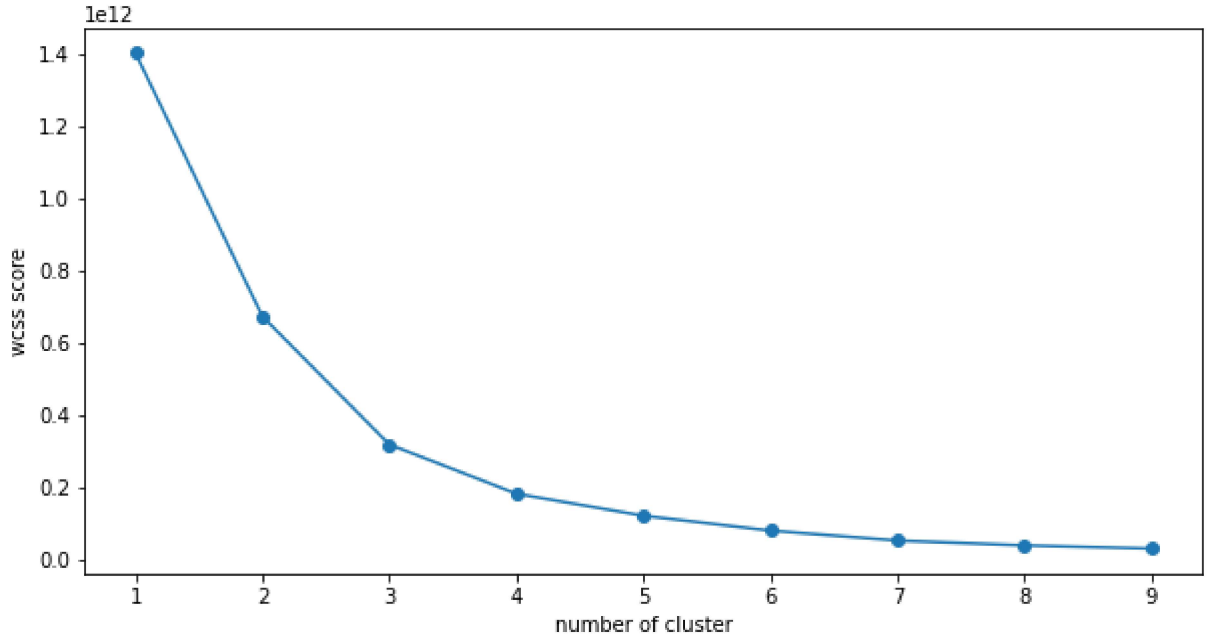
	0	1	2
0	5906.147280	-915.635431	1.367594
1	-5911.774259	485.648253	-3.699403
2	19365.974464	140.756546	3.921406
3	-25606.838986	144.613853	-1.270508
4	6042.016524	281.695662	5.476505
...	...	...	...
2211	8986.337937	-590.284703	-3.469384
2212	11762.637071	351.170032	8.317024
2213	4743.281414	-558.150217	-1.438181
2214	16999.349138	35.879669	5.366465
2215	614.710798	444.982378	-0.002728

2216 rows × 3 columns

elbow method to determine optimum number of clusters

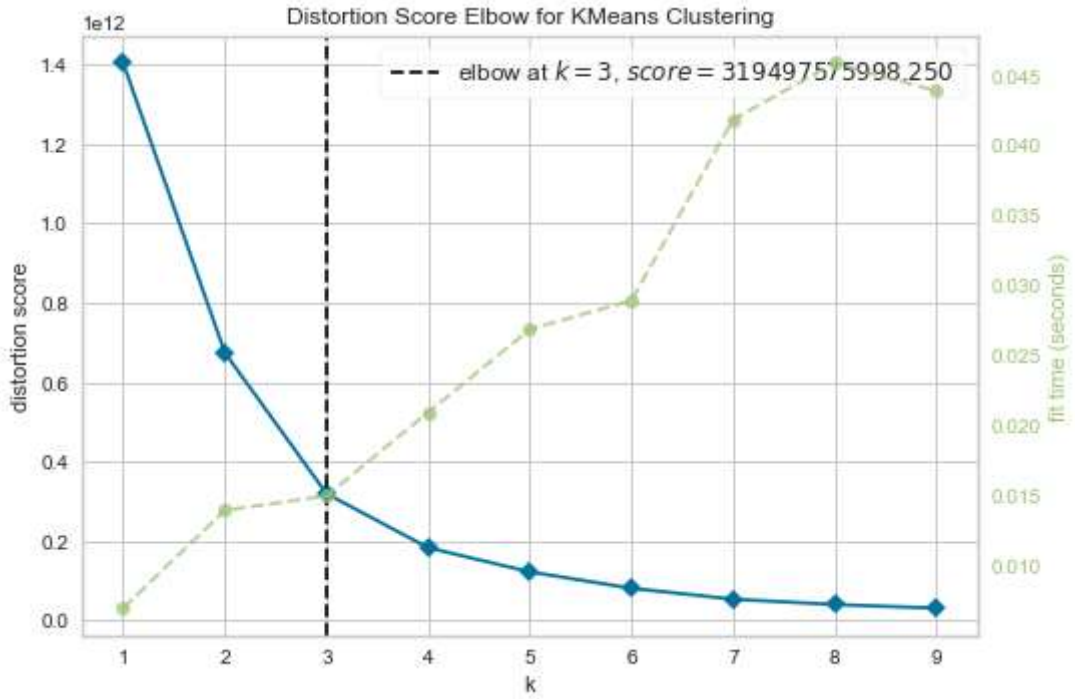
In [34]:
```
wcss = []
for i in range (1,10):
    km=KMeans(n_clusters=i)
    km.fit(pcaX)
    wcss_iter = km.inertia_
    wcss.append(wcss_iter)
```

In [35]:
```
fig,ax = plt.subplots(figsize = (10,5))
plt.plot(range(1,10),wcss, marker = 'o')
plt.xlabel('number of cluster')
plt.ylabel('wcss score')
plt.show()
```



Using yellow bricks to validate the optimum number of clusters

In [36]:
```
from yellowbrick.cluster import KElbowVisualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,10)).fit(pcaX )
visualizer.show()
```



Out[36]: <AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>

Predicting the clusters

In [37]:
```
km = KMeans(n_clusters=3, random_state=42)
predict = km.fit_predict(pcaX)
(predict)
```

Out[37]:
```
array([0, 1, 0, ..., 0, 0, 0])
```

In [38]:
```
km = KMeans(n_clusters=3, random_state=42)
km.fit_transform(pcaX)
final = pd.concat([pcaX,pd.DataFrame({"clusters" : km.labels_})],axis =1)
final
```

Out[38]:

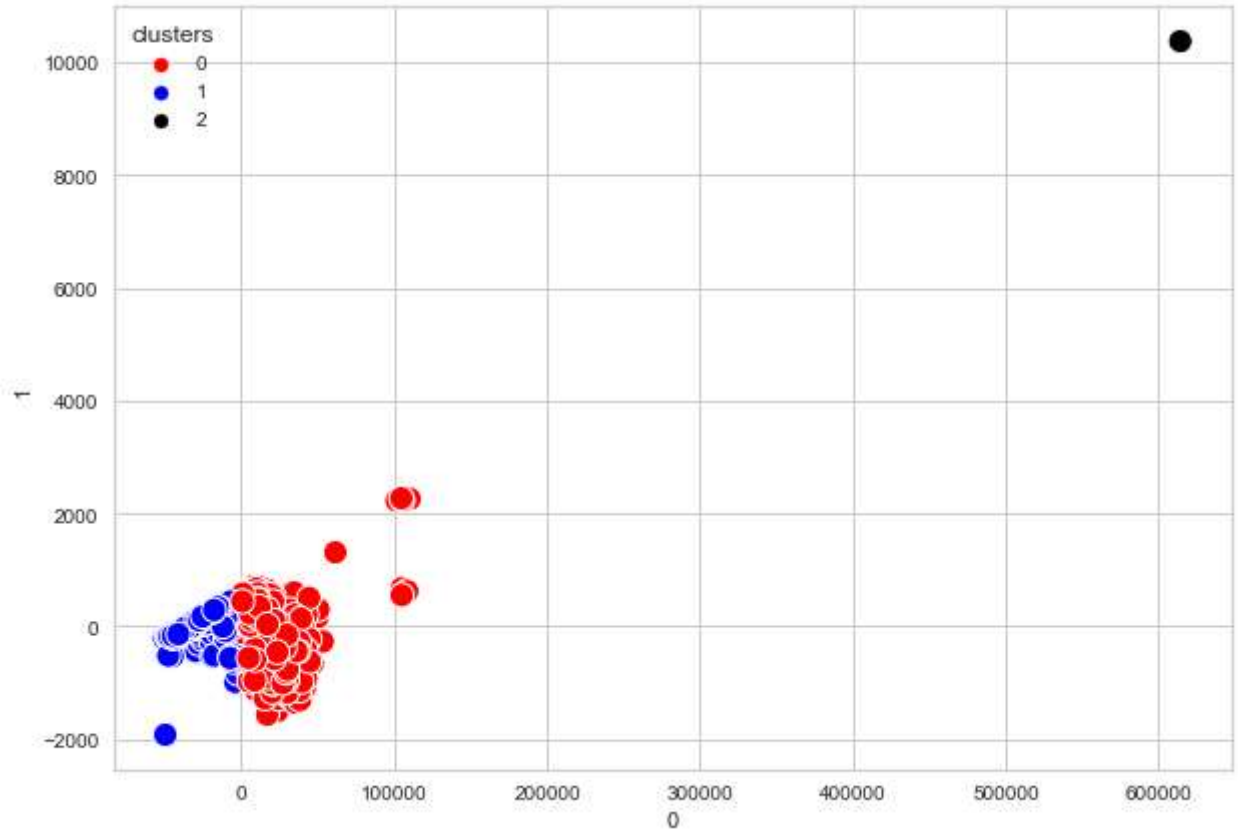
	0	1	2	clusters
0	5906.147280	-915.635431	1.367594	0
1	-5911.774259	485.648253	-3.699403	1
2	19365.974464	140.756546	3.921406	0
3	-25606.838986	144.613853	-1.270508	1
4	6042.016524	281.695662	5.476505	0
...	...	...	...	...
2211	8986.337937	-590.284703	-3.469384	0
2212	11762.637071	351.170032	8.317024	0
2213	4743.281414	-558.150217	-1.438181	0
2214	16999.349138	35.879669	5.366465	0
2215	614.710798	444.982378	-0.002728	0

2216 rows × 4 columns

Plotting the clusters

In [39]:
```
plt.subplots(figsize =(10,7))
sns.scatterplot (data = final , x= 0, y= 1, palette=  ['red','blue', 'black'], hue = 'clusters', s = 150)
```

Out[39]: <AxesSubplot:xlabel='0', ylabel='1'>



In [40]:
```
scaledX_inverse = pd.DataFrame(mc.inverse_transform(scaledX))
scaledX_inverse
```

Out[40]:

	0	1	2	3	4
0	58138.0	7.0	25.0	1617.0	0.0
1	46344.0	5.0	6.0	27.0	2.0
2	71613.0	4.0	21.0	776.0	0.0
3	26646.0	6.0	8.0	53.0	1.0
4	58293.0	5.0	19.0	422.0	1.0
...	...	...	...	...	...
2211	61223.0	5.0	18.0	1341.0	1.0
2212	64014.0	7.0	22.0	444.0	3.0
2213	56981.0	6.0	19.0	1241.0	0.0
2214	69245.0	3.0	23.0	843.0	1.0
2215	52869.0	7.0	11.0	172.0	2.0

2216 rows × 5 columns

In [41]:
```
group_pca = pd.concat([df3.reset_index(), final.clusters],axis =1).drop('index',axis =1)
group_pca
```

Out[41]:

	ID	Education	Income	Recency	NumWebVisitsMonth	num_total	mnt_total	age	marital_class	total_kids	clusters
0	5524	2	58138.0	58	7	25	1617	65	1	0	0
1	2174	2	46344.0	38	5	6	27	68	1	2	1
2	4141	2	71613.0	26	4	21	776	57	0	0	0
3	6182	2	26646.0	26	6	8	53	38	0	1	1
4	5324	4	58293.0	94	5	19	422	41	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
2211	10870	2	61223.0	46	5	18	1341	55	0	1	0
2212	4001	4	64014.0	56	7	22	444	76	0	3	0
2213	7270	2	56981.0	91	6	19	1241	41	1	0	0
2214	8235	3	69245.0	8	3	23	843	66	0	1	0
2215	9405	4	52869.0	40	7	11	172	68	0	2	0

2216 rows × 11 columns

Trying out with TSNE

In [42]:
```
tsne = TSNE(n_components=2, random_state= 42, verbose=3)
tsneX = tsne.fit_transform(scaledX)
```

```
C:\Users\godwi\anaconda3\envs\ML\lib\site-packages\sklearn\manifold\_t_sne.py:800: FutureWarning: The defau
lt initialization in TSNE will change from 'random' to 'pca' in 1.2.
  warnings.warn(
C:\Users\godwi\anaconda3\envs\ML\lib\site-packages\sklearn\manifold\_t_sne.py:810: FutureWarning: The defau
lt learning rate in TSNE will change from 200.0 to 'auto' in 1.2.
  warnings.warn(

[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 2216 samples in 0.002s...
[t-SNE] Computed neighbors for 2216 samples in 0.065s...
[t-SNE] Computed conditional probabilities for sample 1000 / 2216
[t-SNE] Computed conditional probabilities for sample 2000 / 2216
[t-SNE] Computed conditional probabilities for sample 2216 / 2216
[t-SNE] Mean sigma: 0.034931
[t-SNE] Computed conditional probabilities in 0.066s
[t-SNE] Iteration 50: error = 69.9984665, gradient norm = 0.0988166 (50 iterations in 0.987s)
[t-SNE] Iteration 100: error = 61.0957375, gradient norm = 0.0297444 (50 iterations in 0.626s)
[t-SNE] Iteration 150: error = 59.2795029, gradient norm = 0.0179695 (50 iterations in 0.616s)
[t-SNE] Iteration 200: error = 58.4945107, gradient norm = 0.0120632 (50 iterations in 0.616s)
[t-SNE] Iteration 250: error = 58.0402832, gradient norm = 0.0106805 (50 iterations in 0.564s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 58.040283
[t-SNE] Iteration 300: error = 0.8368104, gradient norm = 0.0009693 (50 iterations in 0.683s)
[t-SNE] Iteration 350: error = 0.5914299, gradient norm = 0.0003614 (50 iterations in 0.707s)
[t-SNE] Iteration 400: error = 0.5135034, gradient norm = 0.0002293 (50 iterations in 0.714s)
[t-SNE] Iteration 450: error = 0.4824732, gradient norm = 0.0001763 (50 iterations in 0.714s)
[t-SNE] Iteration 500: error = 0.4674979, gradient norm = 0.0001491 (50 iterations in 0.712s)
[t-SNE] Iteration 550: error = 0.4592229, gradient norm = 0.0001312 (50 iterations in 0.677s)
[t-SNE] Iteration 600: error = 0.4537871, gradient norm = 0.0001204 (50 iterations in 0.682s)
[t-SNE] Iteration 650: error = 0.4487277, gradient norm = 0.0001081 (50 iterations in 0.689s)
[t-SNE] Iteration 700: error = 0.4440266, gradient norm = 0.0001053 (50 iterations in 0.680s)
[t-SNE] Iteration 750: error = 0.4406740, gradient norm = 0.0000993 (50 iterations in 0.654s)
[t-SNE] Iteration 800: error = 0.4376967, gradient norm = 0.0000953 (50 iterations in 0.659s)
[t-SNE] Iteration 850: error = 0.4354333, gradient norm = 0.0000943 (50 iterations in 0.664s)
[t-SNE] Iteration 900: error = 0.4338987, gradient norm = 0.0000849 (50 iterations in 0.657s)
[t-SNE] Iteration 950: error = 0.4315650, gradient norm = 0.0000887 (50 iterations in 0.656s)
[t-SNE] Iteration 1000: error = 0.4300393, gradient norm = 0.0000795 (50 iterations in 0.659s)
[t-SNE] KL divergence after 1000 iterations: 0.430039
```

In [43]:
```
tsneX = pd.DataFrame(tsneX, columns=['tsne1','tsne2'])
tsneX = pd.concat([tsneX,final.clusters],axis = 1)
tsneX
```

Out[43]:

	tsne1	tsne2	clusters
0	-26.703539	-33.247257	0
1	-61.791782	17.564610	1
2	-11.895851	-47.110191	0
3	48.885406	-18.003819	1
4	31.025066	23.141819	0
...	...	...	...
2211	-1.630843	45.372974	0
2212	-30.124926	45.089470	0
2213	-23.241371	-34.825668	0
2214	28.525305	37.963932	0
2215	-46.438347	11.595510	0

2216 rows × 3 columns

In [44]:
```
km_tsne = KMeans(n_clusters=4, random_state = 32)
km_tsne.fit_transform(tsneX)
final_tsne = pd.concat([tsneX, pd.DataFrame({'tsne_clusters' : km_tsne.labels_})],axis =1)
final_tsne
```

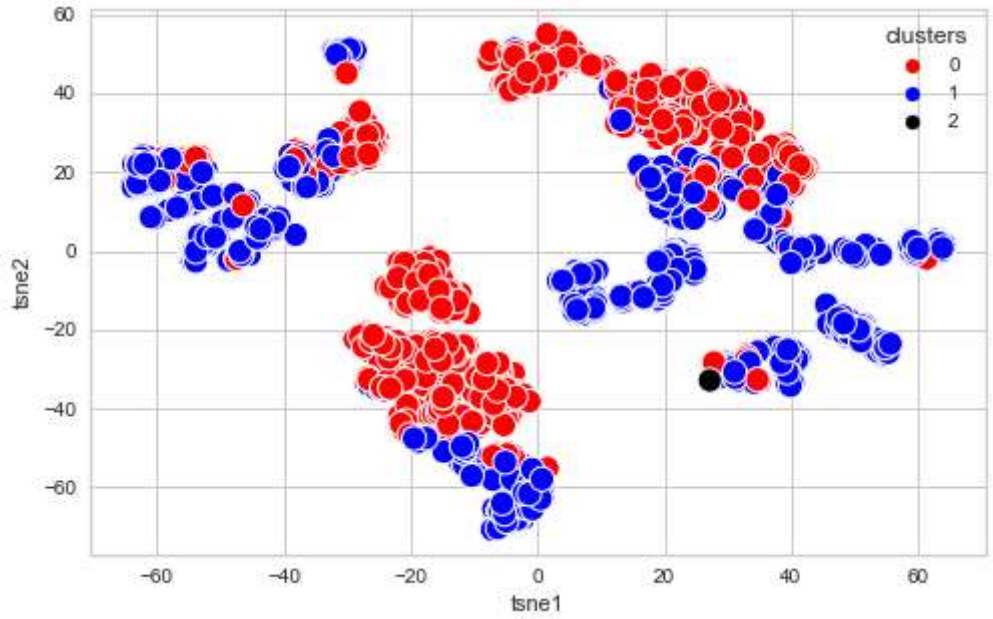
Out[44]:

	tsne1	tsne2	clusters	tsne_clusters
0	-26.703539	-33.247257	0	2
1	-61.791782	17.564610	1	0
2	-11.895851	-47.110191	0	2
3	48.885406	-18.003819	1	1
4	31.025066	23.141819	0	3
...	...	...	...	...
2211	-1.630843	45.372974	0	3
2212	-30.124926	45.089470	0	0
2213	-23.241371	-34.825668	0	2
2214	28.525305	37.963932	0	3
2215	-46.438347	11.595510	0	0

2216 rows × 4 columns

In [45]:
```
plt.subplots(figsize =(8,5))
sns.scatterplot (data = final_tsne , x= 'tsne1', y='tsne2', palette= \
    ['red','blue', 'black'], hue = 'clusters', s = 150)
```

Out[45]: <AxesSubplot:xlabel='tsne1', ylabel='tsne2'>



In [46]:
```
group_tsne = pd.concat ([df3.reset_index(), tsneX.clusters],axis =1).drop('index', axis =1)
group_tsne[group_tsne.clusters==3]
```

Out[46]:

ID	Education	Income	Recency	NumWebVisitsMonth	num_total	mnt_total	age	marital_class	total_kids	clusters

In [47]:
```
group_pca.head()
```

Out[47]:

	ID	Education	Income	Recency	NumWebVisitsMonth	num_total	mnt_total	age	marital_class	total_kids	clusters
0	5524	2	58138.0	58	7	25	1617	65	1	0	0
1	2174	2	46344.0	38	5	6	27	68	1	2	1
2	4141	2	71613.0	26	4	21	776	57	0	0	0
3	6182	2	26646.0	26	6	8	53	38	0	1	1
4	5324	4	58293.0	94	5	19	422	41	0	1	0

In [48]:
```
group_pca.clusters.value_counts()
```

Out[48]:
```
1    1137
0    1078
2       1
Name: clusters, dtype: int64
```

In [49]:
```
group_tsne.clusters.value_counts()
```

Out[49]:
```
1    1137
0    1078
2       1
Name: clusters, dtype: int64
```

In [ ]: