In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
```

In [2]:

```
df=pd.read_csv('sales_data_sample.csv',encoding='unicode_escape')
```

In [3]:

```
df.head()
```

Out[3]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDEF
0	10107	30	95.70	2	2871.00	2/2
1	10121	34	81.35	5	2765.90	5/7/20(
2	10134	41	94.74	2	3884.34	7/1/20(
3	10145	45	83.26	6	3746.70	8/2
4	10159	49	100.00	14	5205.27	10/1

5 rows × 25 columns

In [4]:

```
df.size
```

Out[4]:

```
70575
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ORDERNUMBER       2823 non-null   int64  
 1   QUANTITYORDERED   2823 non-null   int64  
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64  
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object 
 6   STATUS            2823 non-null   object 
 7   QTR_ID            2823 non-null   int64  
 8   MONTH_ID          2823 non-null   int64  
 9   YEAR_ID           2823 non-null   int64  
 10  PRODUCTLINE       2823 non-null   object 
 11  MSRP              2823 non-null   int64  
 12  PRODUCTCODE       2823 non-null   object 
 13  CUSTOMERNAME      2823 non-null   object 
 14  PHONE             2823 non-null   object 
 15  ADDRESSLINE1      2823 non-null   object 
 16  ADDRESSLINE2      302 non-null    object 
 17  CITY              2823 non-null   object 
 18  STATE             1337 non-null   object 
 19  POSTALCODE        2747 non-null   object 
 20  COUNTRY           2823 non-null   object 
 21  TERRITORY         1749 non-null   object 
 22  CONTACTLASTNAME   2823 non-null   object 
 23  CONTACTFIRSTNAME  2823 non-null   object 
 24  DEALSIZE          2823 non-null   object 
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

In [6]:

```
df.shape
```

Out[6]:

```
(2823, 25)
```

In [7]:

```
df.describe().transpose()
```

Out[7]:

	count	mean	std	min	25%	50%	75%
ORDERNUMBER	2823.0	10258.725115	92.085478	10100.00	10180.00	10262.0	10333.
QUANTITYORDERED	2823.0	35.092809	9.741443	6.00	27.00	35.0	43.
PRICEEACH	2823.0	83.658544	20.174277	26.88	68.86	95.7	100.
ORDERLINENUMBER	2823.0	6.466171	4.225841	1.00	3.00	6.0	9.
SALES	2823.0	3553.889072	1841.865106	482.13	2203.43	3184.8	4508.
QTR_ID	2823.0	2.717676	1.203878	1.00	2.00	3.0	4.
MONTH_ID	2823.0	7.092455	3.656633	1.00	4.00	8.0	11.
YEAR_ID	2823.0	2003.815090	0.699670	2003.00	2003.00	2004.0	2004.
MSRP	2823.0	100.715551	40.187912	33.00	68.00	99.0	124.

In [8]:

```
df.columns
```

Out[8]:

```
Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER',
       'SALES', 'ORDERDATE', 'STATUS', 'QTR_ID', 'MONTH_ID', 'YEAR_ID',
       'PRODUCTLINE', 'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE',
       'ADDRESSLINE1', 'ADDRESSLINE2', 'CITY', 'STATE', 'POSTALCODE',
       'COUNTRY', 'TERRITORY', 'CONTACTLASTNAME', 'CONTACTFIRSTNAME',
       'DEALSIZE'],
      dtype='object')
```

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
ORDERNUMBER             0
QUANTITYORDERED         0
PRICEEACH               0
ORDERLINENUMBER         0
SALES                   0
ORDERDATE               0
STATUS                  0
QTR_ID                  0
MONTH_ID                0
YEAR_ID                 0
PRODUCTLINE             0
MSRP                    0
PRODUCTCODE             0
CUSTOMERNAME            0
PHONE                   0
ADDRESSLINE1            0
ADDRESSLINE2         2521
CITY                    0
STATE                1486
POSTALCODE             76
COUNTRY                 0
TERRITORY            1074
CONTACTLASTNAME         0
CONTACTFIRSTNAME        0
DEALSIZE                0
dtype: int64
```

In [10]:

```
df.dtypes
```

Out[10]:

```
ORDERNUMBER            int64
QUANTITYORDERED        int64
PRICEEACH            float64
ORDERLINENUMBER        int64
SALES                float64
ORDERDATE             object
STATUS                object
QTR_ID                 int64
MONTH_ID               int64
YEAR_ID                int64
PRODUCTLINE           object
MSRP                   int64
PRODUCTCODE           object
CUSTOMERNAME          object
PHONE                 object
ADDRESSLINE1          object
ADDRESSLINE2          object
CITY                  object
STATE                 object
POSTALCODE            object
COUNTRY               object
TERRITORY             object
CONTACTLASTNAME       object
CONTACTFIRSTNAME      object
DEALSIZE              object
dtype: object
```

In [11]:

```
li=['ORDERNUMBER','STATUS','CUSTOMERNAME','PHONE','ADDRESSLINE1','ADDRESSLINE2','CITY',
'STATE','POSTALCODE','TERRITORY','CONTACTLASTNAME','CONTACTFIRSTNAME']
```

In [12]:

```
df.drop(li,inplace=True,axis=1)
```

In [13]:

```
df.columns
```

Out[13]:

```
Index(['QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER', 'SALES', 'ORDERD
ATE',
       'QTR_ID', 'MONTH_ID', 'YEAR_ID', 'PRODUCTLINE', 'MSRP', 'PRODUCTCOD
E',
       'COUNTRY', 'DEALSIZE'],
      dtype='object')
```

In [14]:

```
#Checking the categorical columns.
```

In [15]:

```
df['COUNTRY'].unique()
```

Out[15]:

```
array(['USA', 'France', 'Norway', 'Australia', 'Finland', 'Austria', 'UK',
       'Spain', 'Sweden', 'Singapore', 'Canada', 'Japan', 'Italy',
       'Denmark', 'Belgium', 'Philippines', 'Germany', 'Switzerland',
       'Ireland'], dtype=object)
```

In [16]:

```
df['DEALSIZE'].unique()
```

Out[16]:

```
array(['Small', 'Medium', 'Large'], dtype=object)
```

In [17]:

```
df['PRODUCTCODE'].unique()
```

Out[17]:

```
array(['S10_1678', 'S10_1949', 'S10_2016', 'S10_4698', 'S10_4757',
       'S10_4962', 'S12_1099', 'S12_1108', 'S12_1666', 'S12_2823',
       'S12_3148', 'S12_3380', 'S12_3891', 'S12_3990', 'S12_4473',
       'S12_4675', 'S18_1097', 'S18_1129', 'S18_1342', 'S18_1367',
       'S18_1589', 'S18_1662', 'S18_1749', 'S18_1889', 'S18_1984',
       'S18_2238', 'S18_2248', 'S18_2319', 'S18_2325', 'S18_2432',
       'S18_2581', 'S18_2625', 'S18_2795', 'S18_2870', 'S18_2949',
       'S18_2957', 'S18_3029', 'S18_3136', 'S18_3140', 'S18_3232',
       'S18_3259', 'S18_3278', 'S18_3320', 'S18_3482', 'S18_3685',
       'S18_3782', 'S18_3856', 'S18_4027', 'S18_4409', 'S18_4522',
       'S18_4600', 'S18_4668', 'S18_4721', 'S18_4933', 'S24_1046',
       'S24_1444', 'S24_1578', 'S24_1628', 'S24_1785', 'S24_1937',
       'S24_2000', 'S24_2011', 'S24_2022', 'S24_2300', 'S24_2360',
       'S24_2766', 'S24_2840', 'S24_2841', 'S24_2887', 'S24_2972',
       'S24_3151', 'S24_3191', 'S24_3371', 'S24_3420', 'S24_3432',
       'S24_3816', 'S24_3856', 'S24_3949', 'S24_3969', 'S24_4048',
       'S24_4258', 'S24_4278', 'S24_4620', 'S32_1268', 'S32_1374',
       'S32_2206', 'S32_2509', 'S32_3207', 'S32_3522', 'S32_4289',
       'S32_4485', 'S50_1341', 'S50_1392', 'S50_1514', 'S50_4713',
       'S700_1138', 'S700_1691', 'S700_1938', 'S700_2047', 'S700_2466',
       'S700_2610', 'S700_2824', 'S700_2834', 'S700_3167', 'S700_3505',
       'S700_3962', 'S700_4002', 'S72_1253', 'S72_3212'], dtype=object)
```

In [18]:

```
df.drop(['COUNTRY','ORDERDATE'],inplace=True,axis=1)
```

In [20]:

```
df['PRODUCTCODE']=pd.Categorical(df['PRODUCTCODE']).codes
```

In [21]:

```
df.columns
```

Out[21]:

```
Index(['QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER', 'SALES', 'QTR_I
D',
       'MONTH_ID', 'YEAR_ID', 'PRODUCTLINE', 'MSRP', 'PRODUCTCODE',
       'DEALSIZE'],
      dtype='object')
```

In [22]:

```
df1=pd.get_dummies(df['DEALSIZE'])
```

In [23]:

```
df1
```

Out[23]:

	Large	Medium	Small
0	0	0	1
1	0	0	1
2	0	1	0
3	0	1	0
4	0	1	0
...	...	...	...
2818	0	0	1
2819	0	1	0
2820	0	1	0
2821	0	0	1
2822	0	1	0

2823 rows × 3 columns

In [24]:

```
df2=pd.get_dummies(df['PRODUCTLINE'])
```

In [25]:

```
df2
```

Out[25]:

	Classic Cars	Motorcycles	Planes	Ships	Trains	Trucks and Buses	Vintage Cars
0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0
2	0	1	0	0	0	0	0
3	0	1	0	0	0	0	0
4	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...
2818	0	0	0	1	0	0	0
2819	0	0	0	1	0	0	0
2820	0	0	0	1	0	0	0
2821	0	0	0	1	0	0	0
2822	0	0	0	1	0	0	0

2823 rows × 7 columns

In [26]:

```
df.drop(['PRODUCTLINE','DEALSIZE'],inplace=True,axis=1)
```

In [27]:

```
df=pd.concat([df,df1,df2],axis=1)
```

In [28]:

```
df.columns
```

Out[28]:

```
Index(['QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER', 'SALES', 'QTR_I
D',
       'MONTH_ID', 'YEAR_ID', 'MSRP', 'PRODUCTCODE', 'Large', 'Medium',
       'Small', 'Classic Cars', 'Motorcycles', 'Planes', 'Ships', 'Train
s',
       'Trucks and Buses', 'Vintage Cars'],
      dtype='object')
```

In [29]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   QUANTITYORDERED  2823 non-null   int64  
 1   PRICEEACH        2823 non-null   float64
 2   ORDERLINENUMBER  2823 non-null   int64  
 3   SALES            2823 non-null   float64
 4   QTR_ID           2823 non-null   int64  
 5   MONTH_ID         2823 non-null   int64  
 6   YEAR_ID          2823 non-null   int64  
 7   MSRP             2823 non-null   int64  
 8   PRODUCTCODE      2823 non-null   int8   
 9   Large            2823 non-null   uint8  
 10  Medium           2823 non-null   uint8  
 11  Small            2823 non-null   uint8  
 12  Classic Cars     2823 non-null   uint8  
 13  Motorcycles      2823 non-null   uint8  
 14  Planes           2823 non-null   uint8  
 15  Ships            2823 non-null   uint8  
 16  Trains           2823 non-null   uint8  
 17  Trucks and Buses 2823 non-null   uint8  
 18  Vintage Cars     2823 non-null   uint8  
dtypes: float64(2), int64(6), int8(1), uint8(10)
memory usage: 206.9 KB
```

In [30]:

```
df.head()
```

Out[30]:

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	QTR_ID	MONTH_ID	YE
0	30	95.70	2	2871.00	1	2	
1	34	81.35	5	2765.90	2	5	
2	41	94.74	2	3884.34	3	7	
3	45	83.26	6	3746.70	3	8	
4	49	100.00	14	5205.27	4	10	

In [31]:

```
df['PRODUCTCODE'].unique()
```

Out[31]:

```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108], dtype=int8)
```

In [32]:

```
df.columns
```

Out[32]:

```
Index(['QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER', 'SALES', 'QTR_I
D',
       'MONTH_ID', 'YEAR_ID', 'MSRP', 'PRODUCTCODE', 'Large', 'Medium',
       'Small', 'Classic Cars', 'Motorcycles', 'Planes', 'Ships', 'Train
s',
       'Trucks and Buses', 'Vintage Cars'],
      dtype='object')
```

In [33]:

```
from sklearn.cluster import k_means,KMeans
from sklearn.decomposition import PCA
```

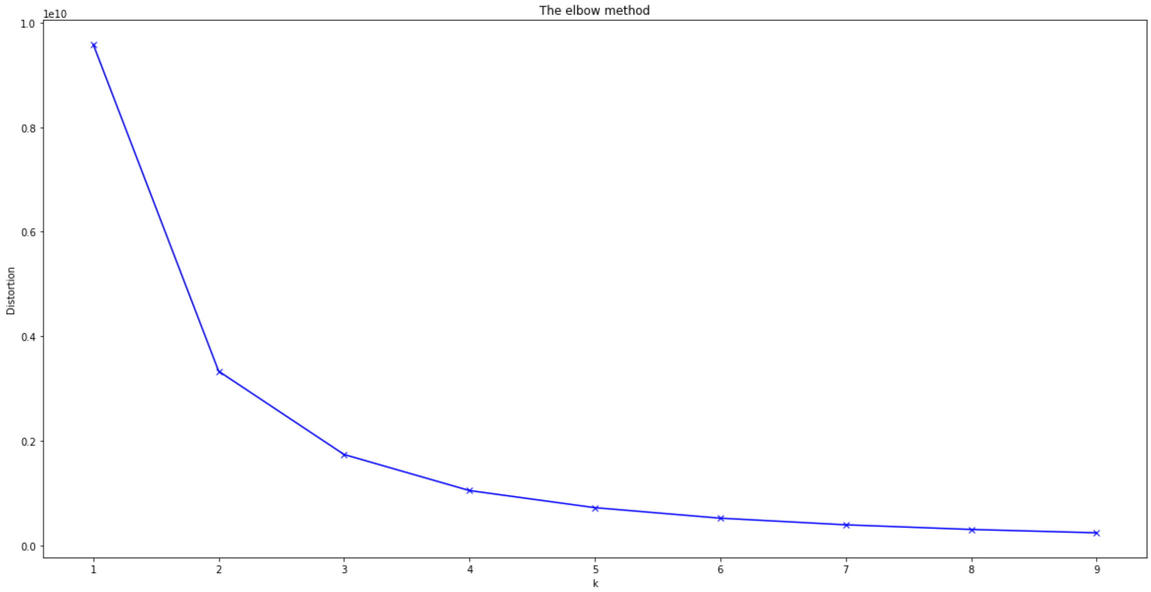
Plotting the Elbow Plot to determine the number of clusters.

In [34]:

```
distotions=[]#within clusters sum of squares from the centroid
k=range(1,10)
for i in k:
    kmeanModel=KMeans(n_clusters=i)
    kmeanModel.fit(df)
    distotions.append(kmeanModel.inertia_) #appending inertia to list
```

In [36]:

```
plt.figure(figsize=(20,10))
plt.plot(k,distotions,'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The elbow method')
plt.show()
```



In [37]:

```
x_train=df.values
```

In [38]:

```
x_train
```

Out[38]:

```
array([[ 30.  ,  95.7 ,   2.  , ...,   0.  ,   0.  ,   0.  ],
       [ 34.  ,  81.35,   5.  , ...,   0.  ,   0.  ,   0.  ],
       [ 41.  ,  94.74,   2.  , ...,   0.  ,   0.  ,   0.  ],
       ...,
       [ 43.  , 100.  ,   4.  , ...,   0.  ,   0.  ,   0.  ],
       [ 34.  ,  62.24,   1.  , ...,   0.  ,   0.  ,   0.  ],
       [ 47.  ,  65.52,   9.  , ...,   0.  ,   0.  ,   0.  ]])
```

In [39]:

```
x_train.shape
```

Out[39]:

```
(2823, 19)
```

In [40]:

```
model=KMeans(n_clusters=3,random_state=2)
```

In [41]:

```
model.fit(x_train)
```

Out[41]:

```
KMeans(n_clusters=3, random_state=2)
```

In [42]:

```
pred=model.predict(x_train)
```

In [43]:

```
unique,counts=np.unique(pred,return_counts=True)
```

In [44]:

```
counts=counts.reshape(1,3)
```

In [45]:

```
counts_df=pd.DataFrame(counts,columns=['Cluster1','Cluster2','Cluster3'])
```

In [46]:

```
counts_df
```

Out[46]:

	Cluster1	Cluster2	Cluster3
0	1083	1367	373

In [47]:

```
unique
```

Out[47]:

```
array([0, 1, 2])
```

In [48]:

```
pca = PCA(n_components=2) #Converting all the features into 2 columns to make it easy t
o visualize using Principal COmponent Analysis.
```

In [50]:

```
reduced_X = pd.DataFrame(pca.fit_transform(x_train),columns=['PCA1','PCA2']) #Creating
 a DataFrame.
```

In [51]:

```
reduced_X.head()
```

Out[51]:

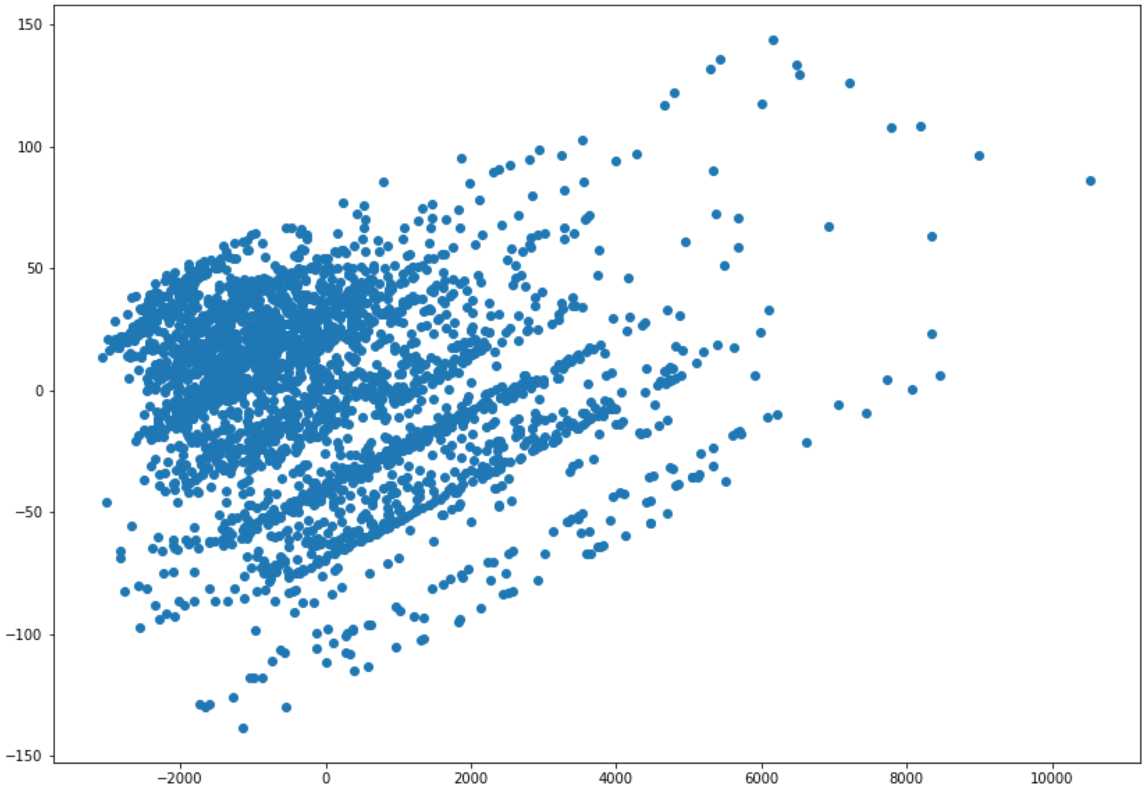
	PCA1	PCA2
0	-682.488323	-42.819535
1	-787.665502	-41.694991
2	330.732170	-26.481208
3	193.040232	-26.285766
4	1651.532874	-6.891196

In [52]:

```
#Plotting the normal Scatter Plot
plt.figure(figsize=(14,10))
plt.scatter(reduced_X['PCA1'],reduced_X['PCA2'])
```

Out[52]:

```
<matplotlib.collections.PathCollection at 0x29847b8aa00>
```



In [53]:

```
model.cluster_centers_ #Finding the centriods. (3 Centriods in total. Each Array contai
ns a centroids for particular feature )
```

Out[53]:

```
array([[ 3.72031394e+01,  9.52120960e+01,  6.44967682e+00,
         4.13868425e+03,  2.72022161e+00,  7.09879963e+00,
         2.00379409e+03,  1.13248384e+02,  5.04469067e+01,
         2.08166817e-17,  1.00000000e+00, -6.66133815e-16,
         3.74884580e-01,  1.15420129e-01,  9.41828255e-02,
         8.21791320e-02,  1.84672207e-02,  1.16343490e-01,
         1.98522622e-01],
       [ 3.08302853e+01,  7.00755230e+01,  6.67300658e+00,
         2.12409474e+03,  2.71762985e+00,  7.09509876e+00,
         2.00381127e+03,  7.84784199e+01,  6.24871982e+01,
         6.93889390e-18,  6.21799561e-02,  9.37820044e-01,
         2.64813460e-01,  1.21433797e-01,  1.29480614e-01,
         1.00219459e-01,  3.87710315e-02,  9.21726408e-02,
         2.53108998e-01],
       [ 4.45871314e+01,  9.98931099e+01,  5.75603217e+00,
         7.09596863e+03,  2.71045576e+00,  7.06434316e+00,
         2.00389008e+03,  1.45823056e+02,  3.14959786e+01,
         4.20911528e-01,  5.79088472e-01,  1.66533454e-16,
         5.33512064e-01,  1.07238606e-01,  7.23860590e-02,
         2.14477212e-02,  1.07238606e-02,  1.31367292e-01,
         1.23324397e-01]])
```

In [54]:

```
reduced_centers = pca.transform(model.cluster_centers_) #Transforming the centroids int
o 3 in x and y coordinates
```

In [55]:

```
reduced_centers
```

Out[55]:

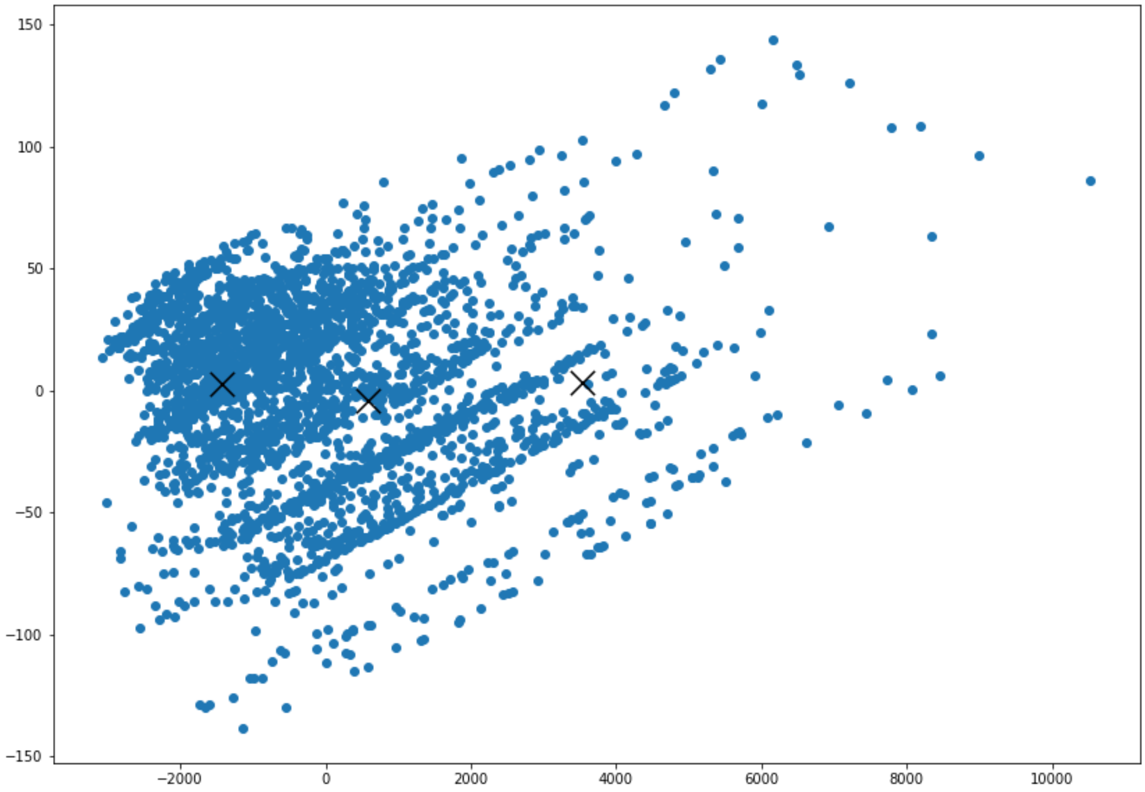
```
array([[ 5.84994044e+02, -4.36786931e+00],
       [-1.43005891e+03,  2.60041009e+00],
       [ 3.54247180e+03,  3.15185487e+00]])
```

In [56]:

```
plt.figure(figsize=(14,10))
plt.scatter(reduced_X['PCA1'],reduced_X['PCA2'])
plt.scatter(reduced_centers[:,0],reduced_centers[:,1],color='black',marker='x',s=300) #
Plotting the centriods
```

Out[56]:

```
<matplotlib.collections.PathCollection at 0x29845f81df0>
```



In [57]:

```
reduced_X['Clusters'] = pred #Adding the Clusters to the reduced dataframe.
```

In [58]:

```
reduced_X.head()
```

Out[58]:

	PCA1	PCA2	Clusters
0	-682.488323	-42.819535	1
1	-787.665502	-41.694991	1
2	330.732170	-26.481208	0
3	193.040232	-26.285766	0
4	1651.532874	-6.891196	0

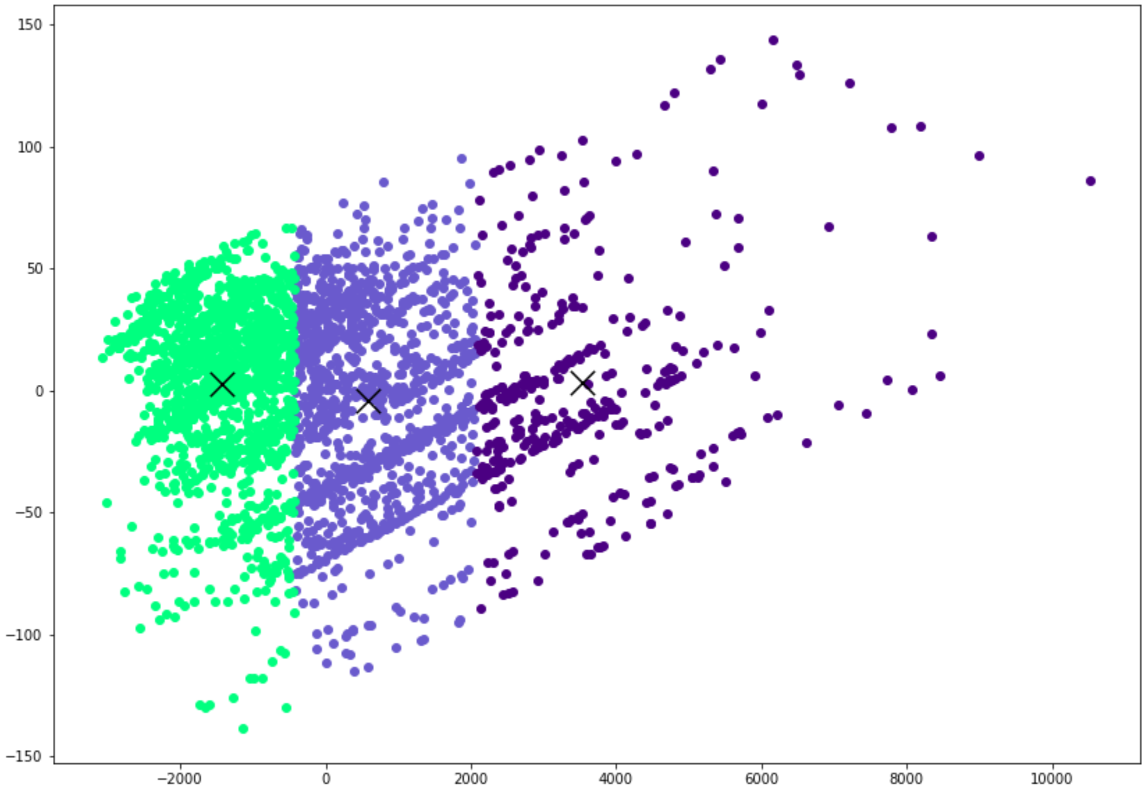
In [59]:

```
#Plotting the clusters
plt.figure(figsize=(14,10))
#                      taking the cluster number and first column           taking the same cluster number and second column      Assigning the color
plt.scatter(reduced_X[reduced_X['Clusters'] == 0].loc[:,'PCA1'],reduced_X[reduced_X['Clusters'] == 0].loc[:,'PCA2'],color='slateblue')
plt.scatter(reduced_X[reduced_X['Clusters'] == 1].loc[:,'PCA1'],reduced_X[reduced_X['Clusters'] == 1].loc[:,'PCA2'],color='springgreen')
plt.scatter(reduced_X[reduced_X['Clusters'] == 2].loc[:,'PCA1'],reduced_X[reduced_X['Clusters'] == 2].loc[:,'PCA2'],color='indigo')


plt.scatter(reduced_centers[:,0],reduced_centers[:,1],color='black',marker='x',s=300)
```

Out[59]:

```
<matplotlib.collections.PathCollection at 0x29833eb4580>
```



In [ ]: