Group B - ASSIGNMENT NO 11

Title - Implement K-means clustering/hierarchial clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method.

```
# Name - Vedant Kulkarni
# Roll Number - 51

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("C:\\Users\\Asus\\Downloads\\
sales_data_sample.csv", encoding='Latin-1')
data.head()

# While utf-8 supports all languages according to pandas'
documentation, utf-8 has a byte structure that must be respected at
all times. Some of the values not included in utf-8 are latin small
letters i with diaeresis, right-pointing double angle quotation mark,
inverted question mark. This are mapped as 0xef, 0xbb and 0xbf bytes
respectively. Hence your error.
```

```
   ORDERNUMBER  QUANTITYORDERED  PRICEEACH  ORDERLINENUMBER
SALES  \
0        10107               30      95.70               2  2871.00

1        10121               34      81.35               5  2765.90

2        10134               41      94.74               2  3884.34

3        10145               45      83.26               6  3746.70

4        10159               49     100.00              14  5205.27


         ORDERDATE   STATUS  QTR_ID  MONTH_ID  YEAR_ID  ...  \
0   2/24/2003 0:00  Shipped       1         2     2003  ...
1    5/7/2003 0:00  Shipped       2         5     2003  ...
2    7/1/2003 0:00  Shipped       3         7     2003  ...
3   8/25/2003 0:00  Shipped       3         8     2003  ...
4  10/10/2003 0:00  Shipped       4        10     2003  ...

                    ADDRESSLINE1  ADDRESSLINE2           CITY STATE  \
0        897 Long Airport Avenue           NaN            NYC    NY
1             59 rue de l'Abbaye           NaN          Reims   NaN
2  27 rue du Colonel Pierre Avia           NaN          Paris   NaN
```

```
3             78934 Hillside Dr.           NaN       Pasadena    CA
4                7734 Strong St.           NaN  San Francisco    CA

  POSTALCODE COUNTRY TERRITORY CONTACTLASTNAME CONTACTFIRSTNAME
DEALSIZE
0      10022     USA       NaN              Yu             Kwai
Small
1      51100  France      EMEA         Henriot             Paul
Small
2      75508  France      EMEA        Da Cunha            Daniel
Medium
3      90003     USA       NaN           Young             Julie
Medium
4        NaN     USA       NaN           Brown             Julie
Medium

[5 rows x 25 columns]
```

```
data.shape
```

```
(2823, 25)
```

```
# Number of NAN values per column in the dataset
data.isnull().sum()
```

```
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
```

```
data.drop(["ORDERNUMBER", "PRICEEACH", "ORDERDATE", "PHONE",
"ADDRESSLINE1", "ADDRESSLINE2", "CITY", "STATE", "TERRITORY",
"POSTALCODE", "CONTACTLASTNAME", "CONTACTFIRSTNAME"], axis = 1,
inplace=True)
```

```
data.head()
```

```
   QUANTITYORDERED  ORDERLINENUMBER    SALES   STATUS  QTR_ID
MONTH_ID  \
0               30                2  2871.00  Shipped       1
2
1               34                5  2765.90  Shipped       2
5
2               41                2  3884.34  Shipped       3
7
3               45                6  3746.70  Shipped       3
8
4               49               14  5205.27  Shipped       4
10

   YEAR_ID  PRODUCTLINE  MSRP PRODUCTCODE              CUSTOMERNAME
COUNTRY  \
0     2003  Motorcycles    95    S10_1678         Land of Toys Inc.
USA
1     2003  Motorcycles    95    S10_1678        Reims Collectables
France
2     2003  Motorcycles    95    S10_1678           Lyon Souveniers
France
3     2003  Motorcycles    95    S10_1678         Toys4GrownUps.com
USA
4     2003  Motorcycles    95    S10_1678  Corporate Gift Ideas Co.
USA

  DEALSIZE
0    Small
1    Small
2   Medium
3   Medium
4   Medium
```

```
data.isnull().sum()
```

```
QUANTITYORDERED    0
ORDERLINENUMBER    0
SALES              0
STATUS             0
QTR_ID             0
MONTH_ID           0
```

```
YEAR_ID             0
PRODUCTLINE         0
MSRP                0
PRODUCTCODE         0
CUSTOMERNAME        0
COUNTRY             0
DEALSIZE            0
dtype: int64
```

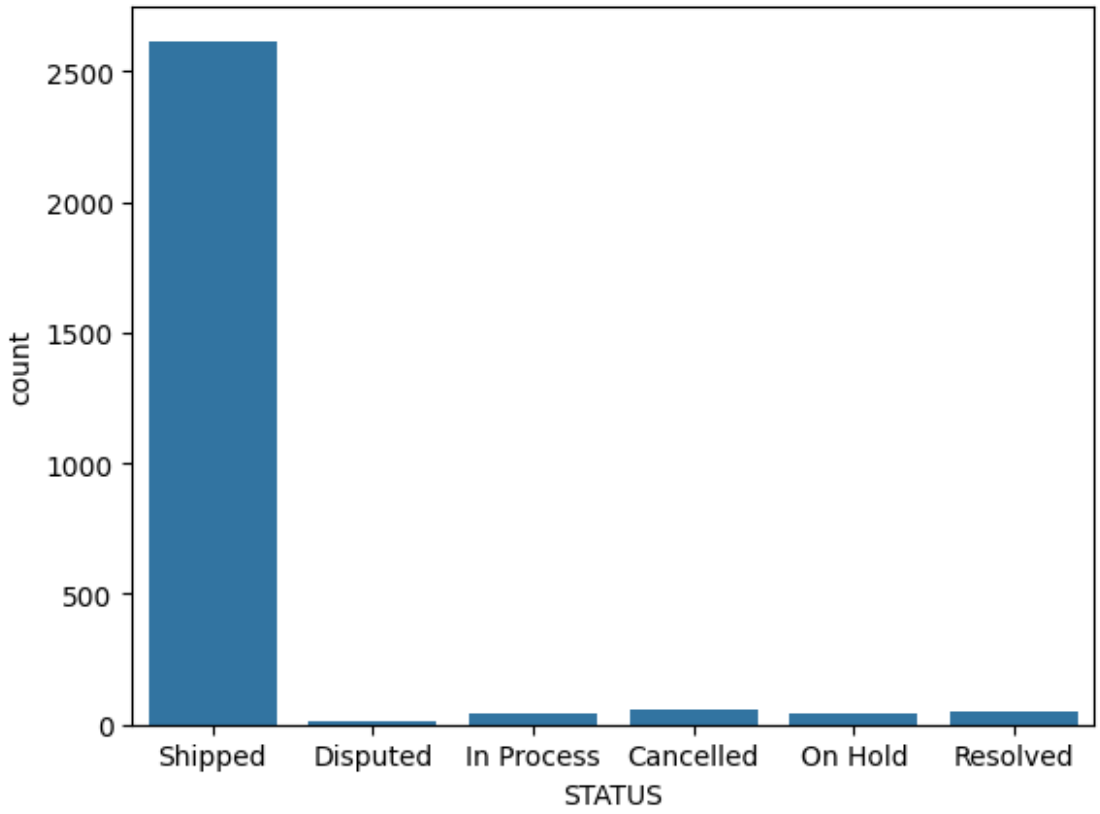
Exploratary Data Analysis

```
data.describe()
```

```
       QUANTITYORDERED  ORDERLINENUMBER         SALES       QTR_ID  \
count      2823.000000      2823.000000   2823.000000  2823.000000
mean         35.092809         6.466171   3553.889072     2.717676
std           9.741443         4.225841   1841.865106     1.203878
min           6.000000         1.000000    482.130000     1.000000
25%          27.000000         3.000000   2203.430000     2.000000
50%          35.000000         6.000000   3184.800000     3.000000
75%          43.000000         9.000000   4508.000000     4.000000
max          97.000000        18.000000  14082.800000     4.000000

          MONTH_ID     YEAR_ID         MSRP
count  2823.000000  2823.00000  2823.000000
mean      7.092455  2003.81509   100.715551
std       3.656633     0.69967    40.187912
min       1.000000  2003.00000    33.000000
25%       4.000000  2003.00000    68.000000
50%       8.000000  2004.00000    99.000000
75%      11.000000  2004.00000   124.000000
max      12.000000  2005.00000   214.000000
```

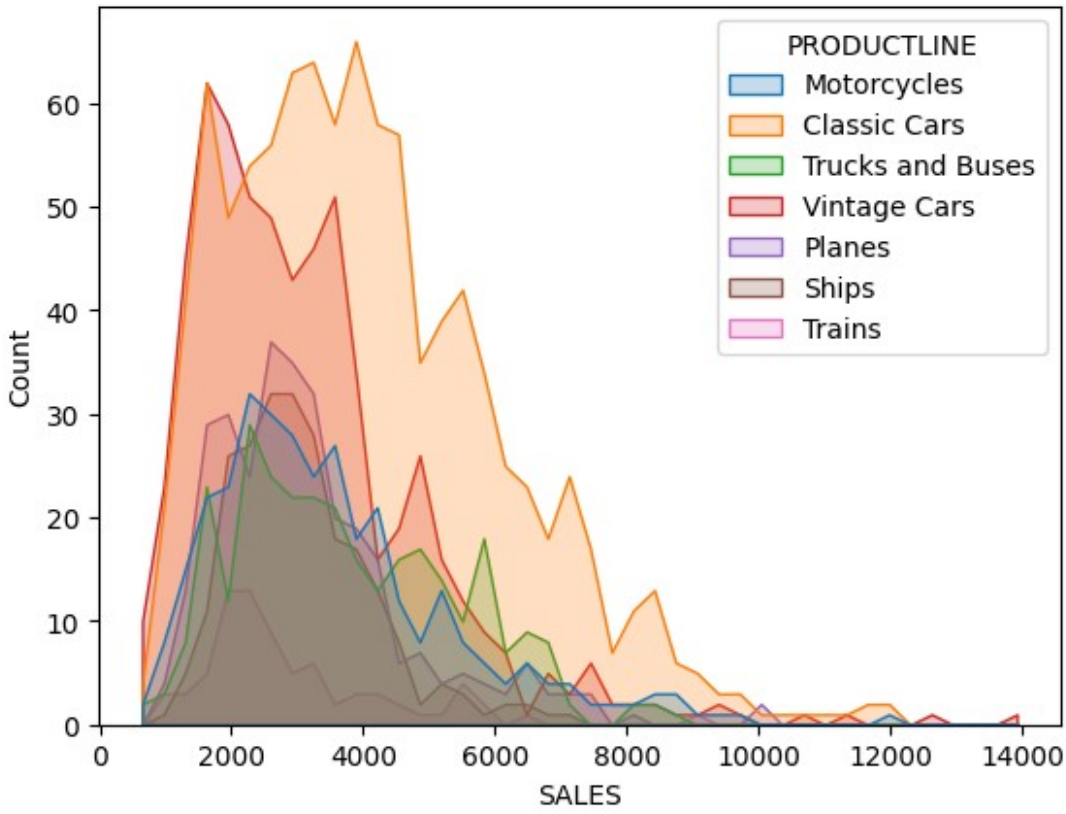
```
sns.countplot(data = data , x = 'STATUS')
```

```
<Axes: xlabel='STATUS', ylabel='count'>
```




```
import seaborn as sns

sns.histplot(x = 'SALES' , hue = 'PRODUCTLINE', data = data,
             element="poly")
```

```
<Axes: xlabel='SALES', ylabel='Count'>
```



Here we can see all the catagory lies in the range of price and hence in this we be creating a cluster on targeting the same

```
data['PRODUCTLINE'].unique()
```

```
array(['Motorcycles', 'Classic Cars', 'Trucks and Buses', 'Vintage
Cars',
       'Planes', 'Ships', 'Trains'], dtype=object)
```

```
#checking the duplicated values
data.drop_duplicates(inplace=True)
```

```
data.info()
```

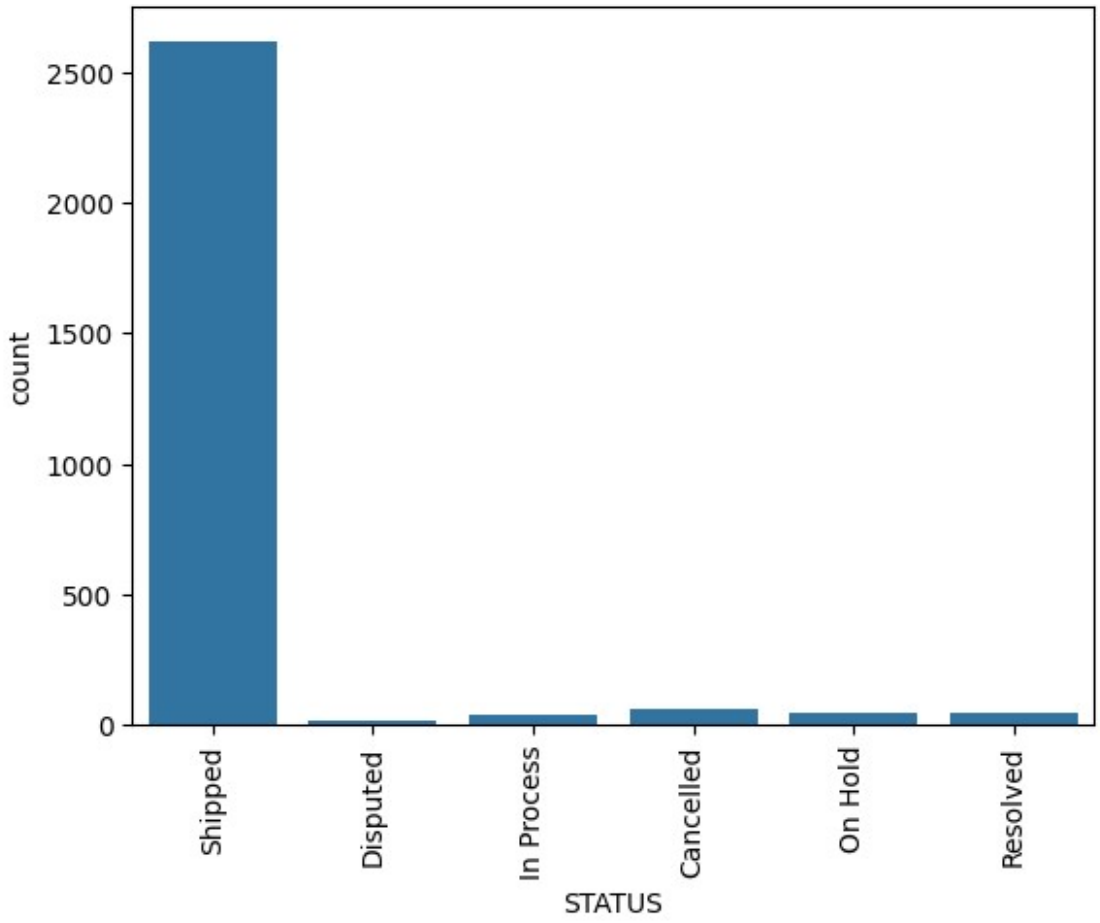
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   QUANTITYORDERED  2823 non-null   int64
 1   ORDERLINENUMBER  2823 non-null   int64
 2   SALES            2823 non-null   float64
 3   STATUS           2823 non-null   object
 4   QTR_ID           2823 non-null   int64
 5   MONTH_ID         2823 non-null   int64
```

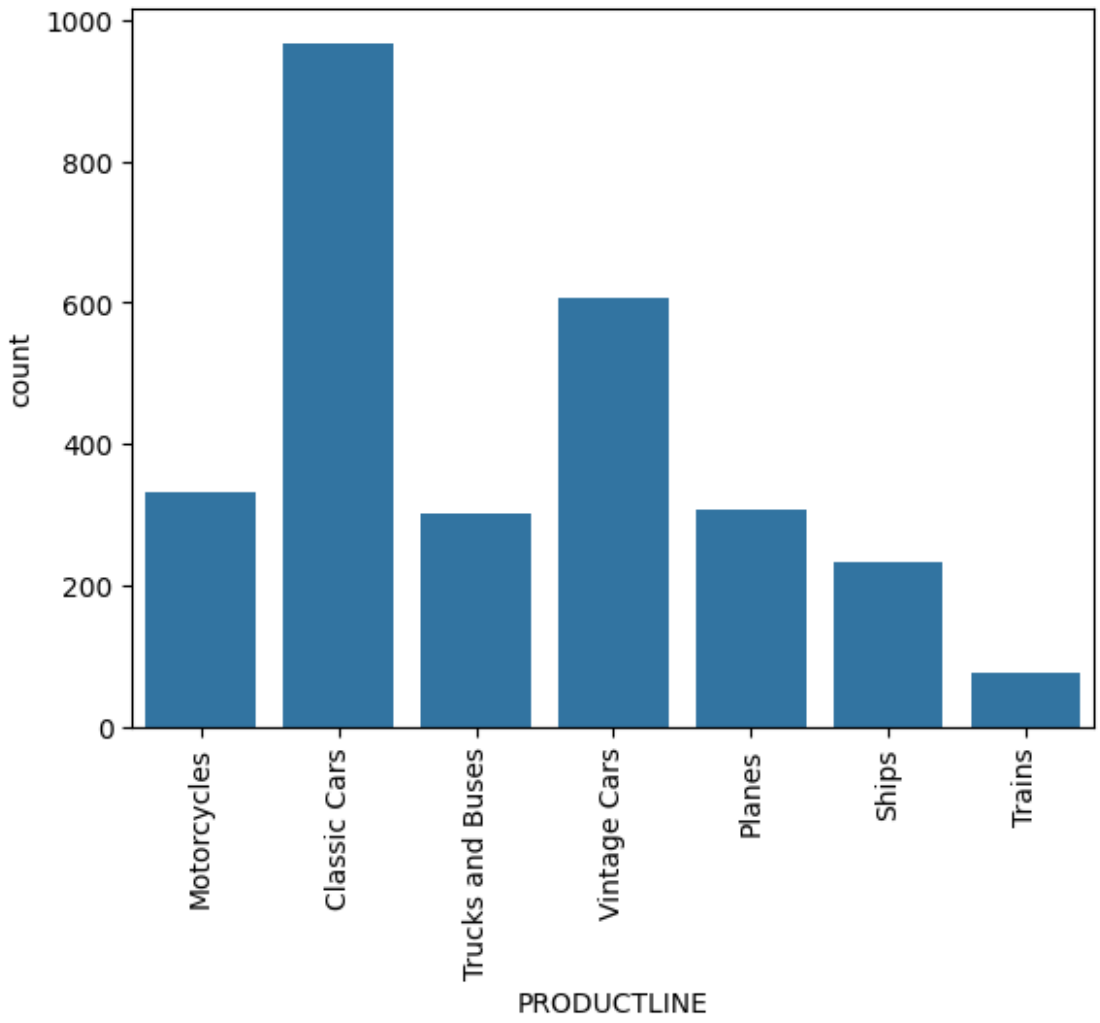
```
 6   YEAR_ID           2823 non-null   int64
 7   PRODUCTLINE       2823 non-null   object
 8   MSRP              2823 non-null   int64
 9   PRODUCTCODE       2823 non-null   object
 10  CUSTOMERNAME      2823 non-null   object
 11  COUNTRY           2823 non-null   object
 12  DEALSIZE          2823 non-null   object
dtypes: float64(1), int64(6), object(6)
memory usage: 286.8+ KB
```

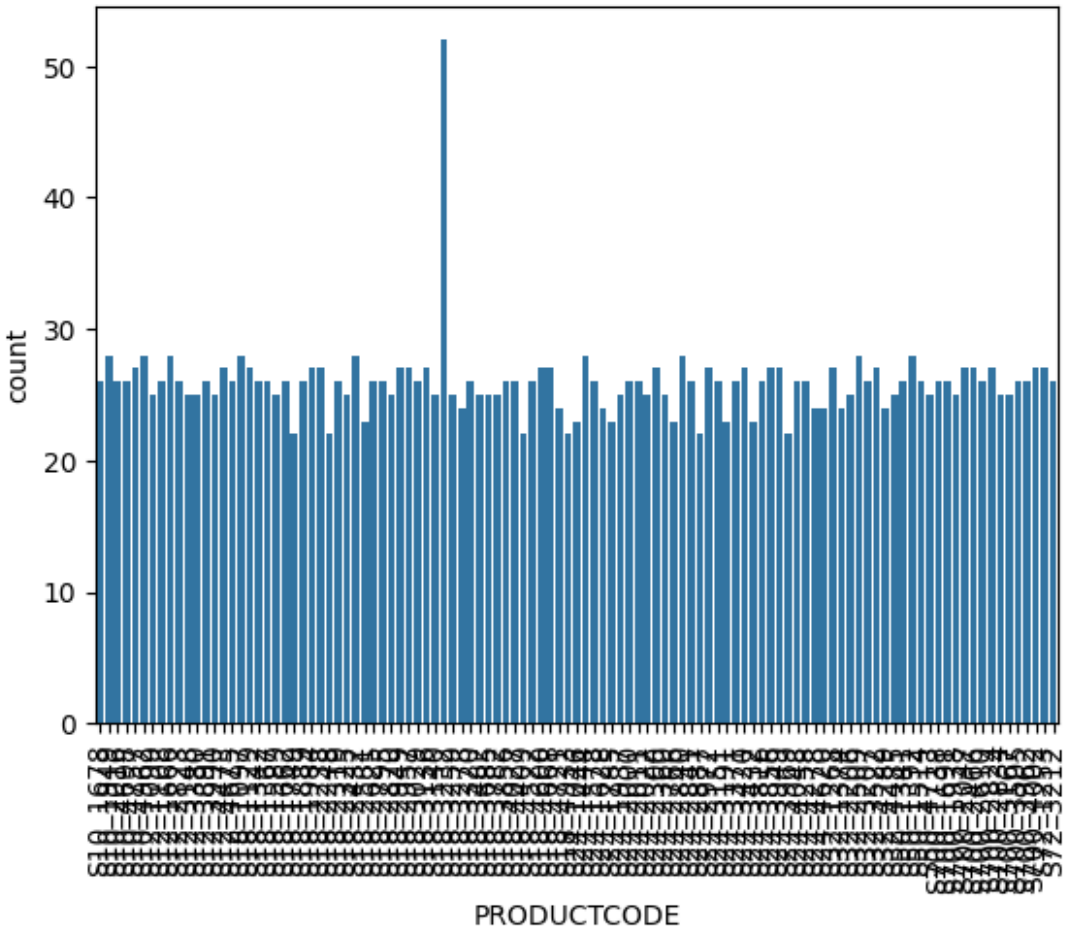
```
list_cat = data.select_dtypes(include=['object']).columns.tolist()

list_cat
```

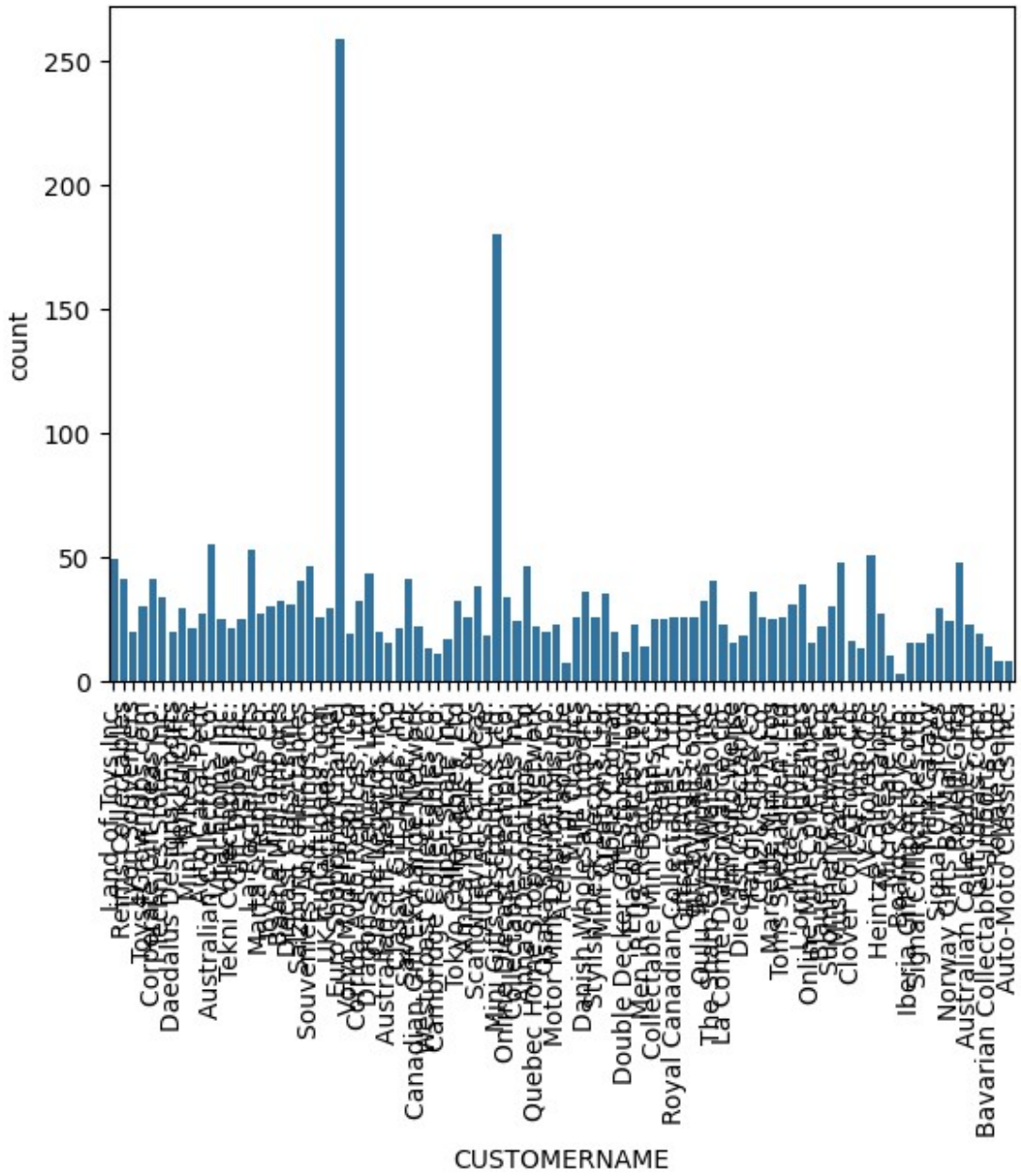
```
['STATUS', 'PRODUCTLINE', 'PRODUCTCODE', 'CUSTOMERNAME', 'COUNTRY',
'DEALSIZE']
```

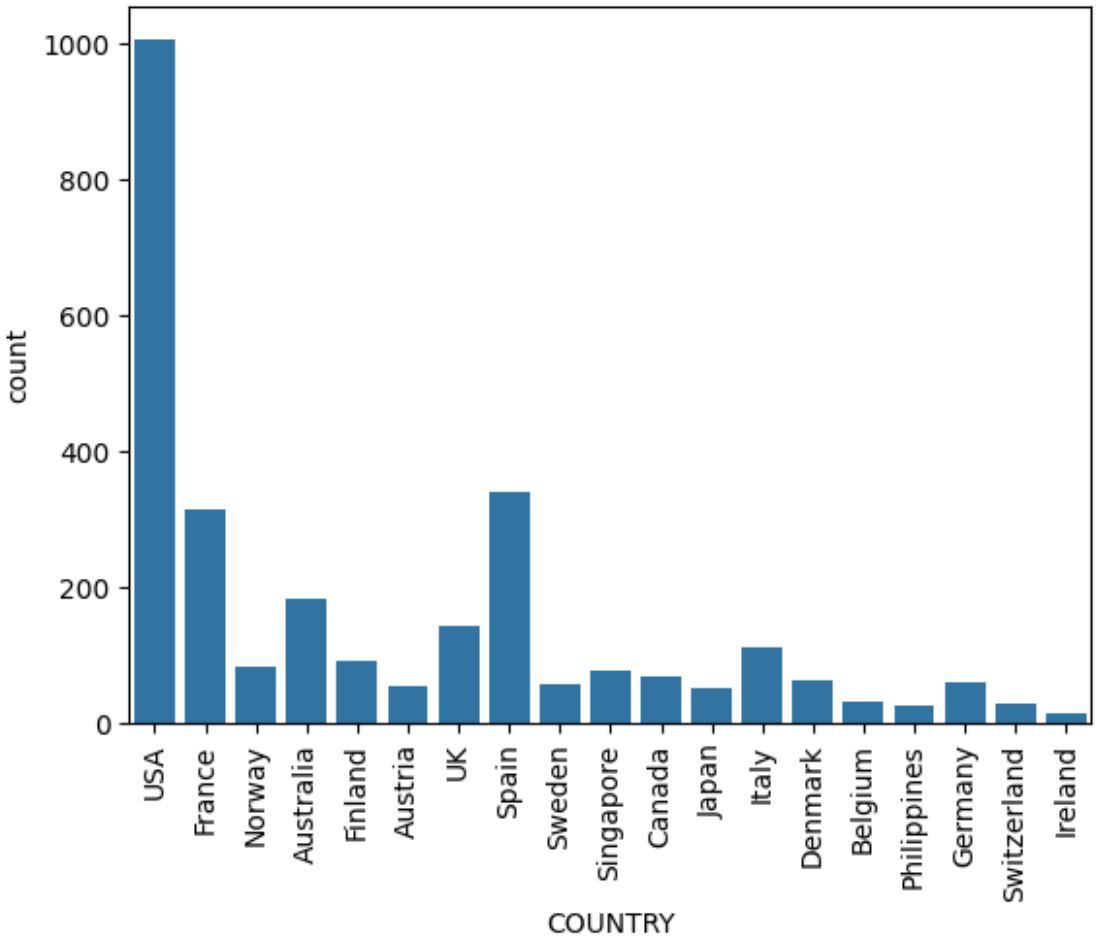
```
for i in list_cat:
  sns.countplot(data = data ,x = i)
  plt.xticks(rotation = 90)
  plt.show()
```

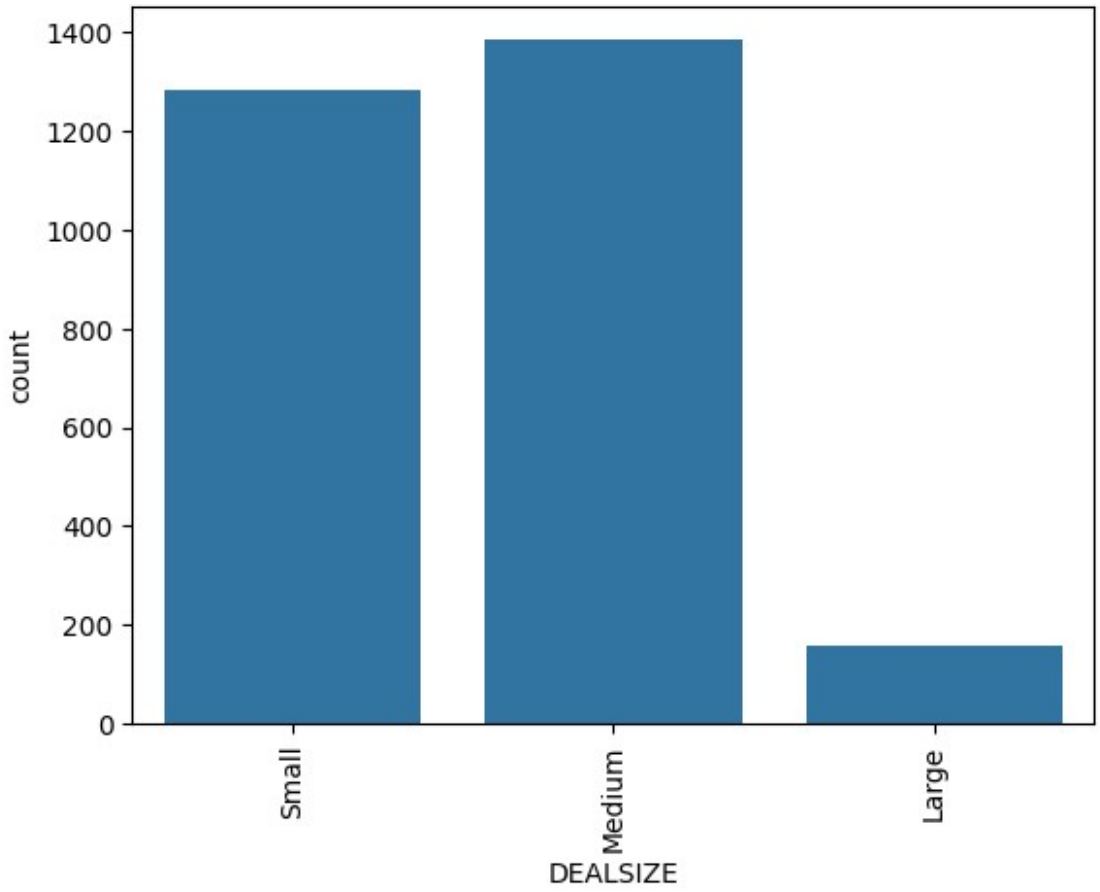



1000
800
600
400
200
0
count
Motorcycles
Classic Cars
Trucks and Buses
Vintage Cars
Planes
Ships
Trains
PRODUCTLINE


50
40
30
20
10
0
count
PRODUCTCODE


250
200
150
100
50
0
count
CUSTOMERNAME


1000
800
600
400
200
0
count
USA
France
Norway
Australia
Finland
Austria
UK
Spain
Sweden
Singapore
Canada
Japan
Italy
Denmark
Belgium
Philippines
Germany
Switzerland
Ireland
COUNTRY




```
#dealing with the catagorical features
from sklearn import preprocessing
le = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
for i in list_cat:
  data[i]= le.fit_transform(data[i])

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   QUANTITYORDERED  2823 non-null   int64
 1   ORDERLINENUMBER  2823 non-null   int64
 2   SALES            2823 non-null   float64
 3   STATUS           2823 non-null   int32
 4   QTR_ID           2823 non-null   int64
 5   MONTH_ID         2823 non-null   int64
 6   YEAR_ID          2823 non-null   int64
```

```
 7   PRODUCTLINE      2823 non-null   int32
 8   MSRP             2823 non-null   int64
 9   PRODUCTCODE      2823 non-null   int32
 10  CUSTOMERNAME     2823 non-null   int32
 11  COUNTRY          2823 non-null   int32
 12  DEALSIZE         2823 non-null   int32
dtypes: float64(1), int32(6), int64(6)
memory usage: 220.7 KB
```

```
data['SALES'] = data['SALES'].astype(int)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   QUANTITYORDERED  2823 non-null   int64
 1   ORDERLINENUMBER  2823 non-null   int64
 2   SALES            2823 non-null   int32
 3   STATUS           2823 non-null   int32
 4   QTR_ID           2823 non-null   int64
 5   MONTH_ID         2823 non-null   int64
 6   YEAR_ID          2823 non-null   int64
 7   PRODUCTLINE      2823 non-null   int32
 8   MSRP             2823 non-null   int64
 9   PRODUCTCODE      2823 non-null   int32
 10  CUSTOMERNAME     2823 non-null   int32
 11  COUNTRY          2823 non-null   int32
 12  DEALSIZE         2823 non-null   int32
dtypes: int32(7), int64(6)
memory usage: 209.6 KB
```

```
data.describe()
```

```
       QUANTITYORDERED  ORDERLINENUMBER         SALES       STATUS  \
count      2823.000000      2823.000000   2823.000000  2823.000000
mean         35.092809         6.466171   3553.421537     4.782501
std           9.741443         4.225841   1841.865754     0.879416
min           6.000000         1.000000    482.000000     0.000000
25%          27.000000         3.000000   2203.000000     5.000000
50%          35.000000         6.000000   3184.000000     5.000000
75%          43.000000         9.000000   4508.000000     5.000000
max          97.000000        18.000000  14082.000000     5.000000

            QTR_ID     MONTH_ID     YEAR_ID  PRODUCTLINE         MSRP
\
count  2823.000000  2823.000000  2823.00000  2823.000000  2823.000000

mean      2.717676     7.092455  2003.81509     2.515055   100.715551
```

```
std       1.203878     3.656633     0.69967     2.411665    40.187912

min       1.000000     1.000000  2003.00000     0.000000    33.000000

25%       2.000000     4.000000  2003.00000     0.000000    68.000000

50%       3.000000     8.000000  2004.00000     2.000000    99.000000

75%       4.000000    11.000000  2004.00000     5.000000   124.000000

max       4.000000    12.000000  2005.00000     6.000000   214.000000
```

```
       PRODUCTCODE  CUSTOMERNAME      COUNTRY     DEALSIZE
count  2823.000000   2823.000000  2823.000000  2823.000000
mean     53.773291     46.212186    12.029401     1.398512
std      31.585298     24.936147     6.169774     0.592498
min       0.000000      0.000000     0.000000     0.000000
25%      27.000000     29.000000     6.000000     1.000000
50%      53.000000     45.000000    14.000000     1.000000
75%      81.000000     67.000000    18.000000     2.000000
max     108.000000     91.000000    18.000000     2.000000
```

```
## taget feature are Sales and productline
X = data[['SALES','PRODUCTCODE']]
```

```
data.columns
```

```
Index(['QUANTITYORDERED', 'ORDERLINENUMBER', 'SALES', 'STATUS',
'QTR_ID',
       'MONTH_ID', 'YEAR_ID', 'PRODUCTLINE', 'MSRP', 'PRODUCTCODE',
       'CUSTOMERNAME', 'COUNTRY', 'DEALSIZE'],
      dtype='object')
```

K Means implementation

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0).fit(X)
```

```
kmeans.labels_
```

```
array([2, 2, 2, ..., 3, 0, 2])
```

```
kmeans.inertia_
```

```
1043164092.8545694
```

```
kmeans.n_iter_
```

```
15
```

```
kmeans.cluster_centers_
```

```
array([[1913.93425926,   63.19907407],
       [8023.78238342,   28.35751295],
       [3489.45517241,   55.50640394],
       [5371.72523364,   40.62616822]])
```

```
#getting the size of the clusters
from collections import Counter
Counter(kmeans.labels_)
```

```
Counter({0: 1078, 2: 1015, 3: 537, 1: 193})
```

Hence the Nmber of Clusters to be choosen Will be 4 according to the elbow method

```
sns.scatterplot(data=X, x="SALES", y="PRODUCTCODE",
hue=kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:,0],
kmeans.cluster_centers_[:,1],
            marker="X", c="r", s=80, label="centroids")
plt.legend()
plt.show()
```