Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method.

Dataset link : https://www.kaggle.com/datasets/kyanyoga/sample-sales-data

▾ Download libraries

```
from google.colab import drive
drive.mount('/content/drive')

     Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import zipfile
import cv2
import plotly.express as px
```

```
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.cluster import KMeans

%matplotlib inline
```

```
df = pd.read_csv('/content/drive/MyDrive/ML/sales_data_sample.csv', encoding = 'unicode
df.head()
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	S
0	10107	30	95.70	2	2871.00	2003-02-24	Sh
1	10121	34	81.35	5	2765.90	2003-05-07	Sh
2	10134	41	94.74	2	3884.34	2003-07-01	Sh
3	10145	45	83.26	6	3746.70	2003-08-25	Sh
4	10159	49	100.00	14	5205.27	2003-10-10	Sh

5 rows × 25 columns

```
df.dtypes
```

```
ORDERNUMBER                  int64
QUANTITYORDERED              int64
PRICEEACH                  float64
ORDERLINENUMBER              int64
SALES                      float64
ORDERDATE           datetime64[ns]
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
```

```
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   datetime64[ns]
 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: datetime64[ns](1), float64(2), int64(7), object(15)
memory usage: 551.5+ KB
```

```
df.isna().mean()
```

```
ORDERNUMBER         0.000000
QUANTITYORDERED     0.000000
PRICEEACH           0.000000
ORDERLINENUMBER     0.000000
SALES               0.000000
ORDERDATE           0.000000
STATUS              0.000000
QTR_ID              0.000000
MONTH_ID            0.000000
YEAR_ID             0.000000
PRODUCTLINE         0.000000
MSRP                0.000000
PRODUCTCODE         0.000000
CUSTOMERNAME        0.000000
PHONE               0.000000
ADDRESSLINE1        0.000000
ADDRESSLINE2        0.893022
CITY                0.000000
STATE               0.526390
POSTALCODE          0.026922
COUNTRY             0.000000
TERRITORY           0.380446
CONTACTLASTNAME     0.000000
CONTACTFIRSTNAME    0.000000
DEALSIZE            0.000000
dtype: float64
```

```
df_drop  = ['ADDRESSLINE1', 'ADDRESSLINE2', 'POSTALCODE', 'CITY', 'TERRITORY', 'PHONE',
df = df.drop(df_drop, axis=1)
df.head(3)
```

	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	STATUS	QTR_ID
0	30	95.70	2	2871.00	2003-02-24	Shipped	1
1	34	81.35	5	2765.90	2003-05-07	Shipped	2

Drop georaphic features and names, phone

```
df.shape
```

```
(2823, 14)
```

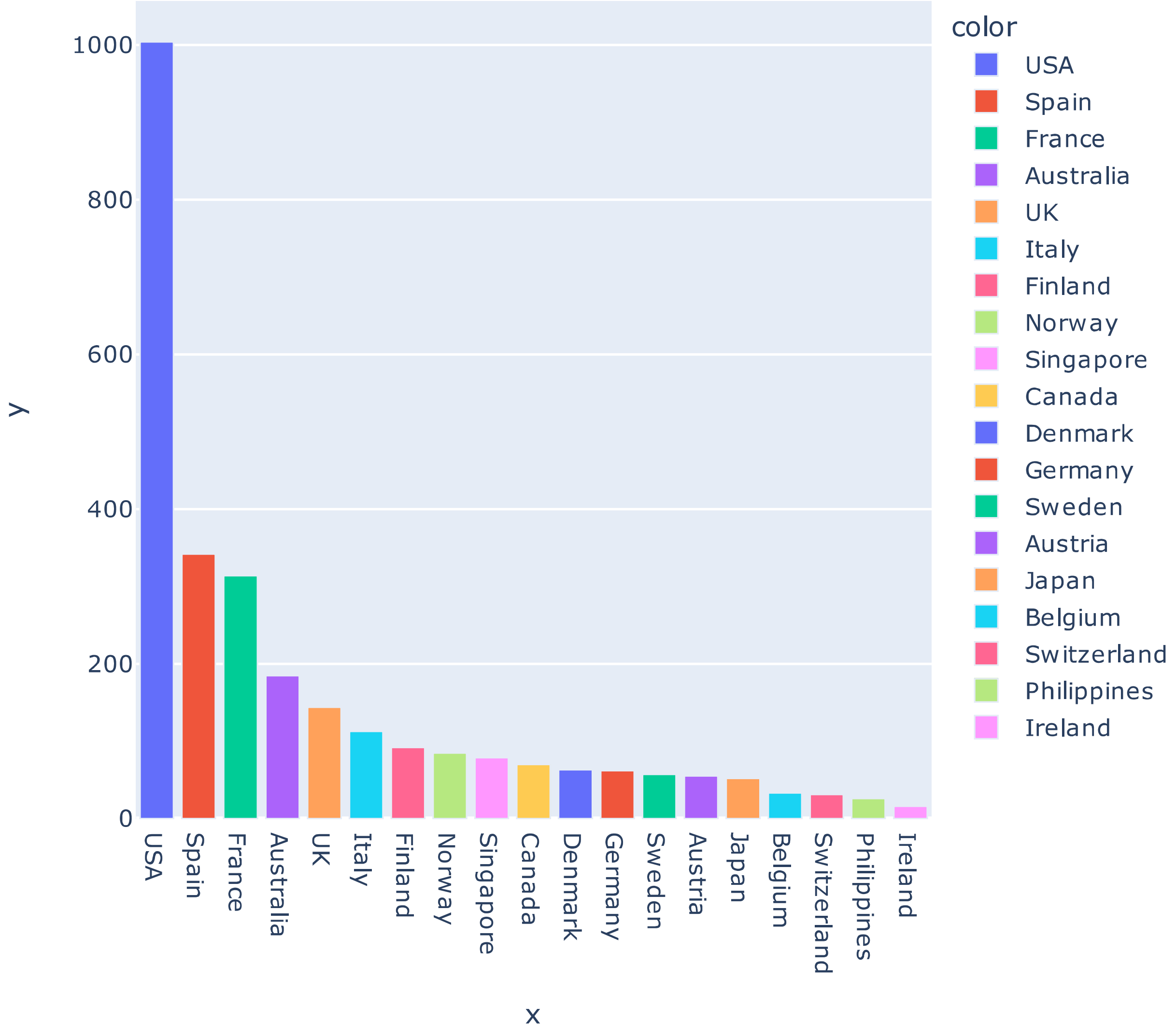
```
df.isna().sum()
```

```
QUANTITYORDERED    0
```

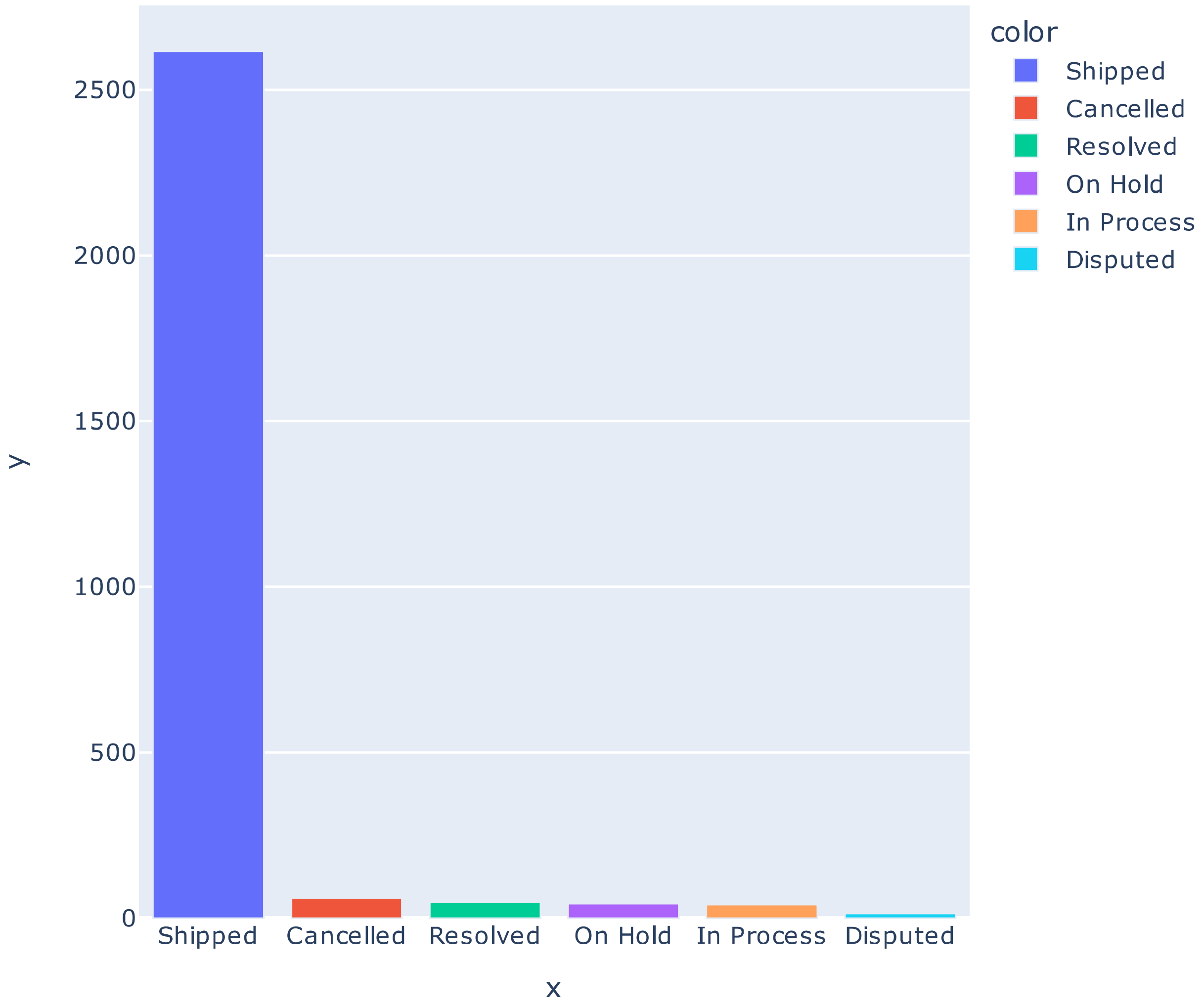
```
     PRICEEACH          0
     ORDERLINENUMBER    0
     SALES              0
     ORDERDATE          0
     STATUS             0
     QTR_ID             0
     MONTH_ID           0
     YEAR_ID            0
     PRODUCTLINE        0
     MSRP               0
     PRODUCTCODE        0
     COUNTRY            0
     DEALSIZE           0
     dtype: int64
```

```
def barplot_visualization(x):
  fig = plt.Figure(figsize = (12, 6))
  fig = px.bar(x = df[x].value_counts().index, y = df[x].value_counts(), color = df[x].
  fig.show();
```

```
barplot_visualization('COUNTRY')
```



```
barplot_visualization('STATUS');
```



Drop unbalanced feature

```
df.drop(columns=['STATUS'], axis=1, inplace=True)
```

```
print('Columns resume: ', df.shape[1])
```

```
    Columns resume:  13
```

```
barplot_visualization('PRODUCTLINE')
```