K MEANS CLUSTERING 4

```
# Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset.
# Determine the number of clusters using the elbow method.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("sales_data_sample.csv", encoding='Latin-1')
data.head()


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


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


data.isnull().sum()


data.describe()


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


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

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

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

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

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

#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()

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

data.describe()

## taget feature are Sales and productline
```

```
X = data[['SALES','PRODUCTCODE']]
data.columns

from yellowbrick.cluster import KElbowVisualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,12)).fit(X)
visualizer.show()

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

kmeans.inertia_

kmeans.n_iter_

kmeans.cluster_centers_

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

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()
```