In [46]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
```

In [47]:
```
df = pd.read_csv('C:/Users/AjithKumar.Pola/Downloads/clustering_dataset.csv')
```

In [48]:
```
df.head()
```

Out[48]:

	CUSTOMER_ID	BALANCE	BALANCE_FREQ	PURCHASES	ONE_OFF_PURCHASE	INSTALLMEI
0	C10001	40.900749	0.818182	95.40	0.00	
1	C10002	3202.467416	0.909091	0.00	0.00	
2	C10003	2495.148862	1.000000	773.17	773.17	
3	C10004	1666.670542	0.636364	1499.00	1499.00	
4	C10005	817.714335	1.000000	16.00	16.00	

In [49]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   CUSTOMER_ID                       8950 non-null   object
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQ                      8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONE_OFF_PURCHASE                  8950 non-null   float64
 5   INSTALLMENT_PURCHASES             8950 non-null   float64
 6   CASH_IN_ADVANCE                   8950 non-null   float64
 7   PURCHASE_FREQ                     8950 non-null   float64
 8   ONE_OFF_PURCHASES_FREQUENCY       8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64
 12  PURCHASES_TRX                     8950 non-null   int64
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

In [50]:
```
df.describe()
```

Out[50]:

	BALANCE	BALANCE_FREQ	PURCHASES	ONE_OFF_PURCHASE	INSTALLMENT_PURCHA
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.00
mean	1564.474828	0.877271	1003.204834	592.437371	411.06
std	2081.531879	0.236904	2136.634782	1659.887917	904.33
min	0.000000	0.000000	0.000000	0.000000	0.00
25%	128.281915	0.888889	39.635000	0.000000	0.00
50%	873.385231	1.000000	361.280000	38.000000	89.00
75%	2054.140036	1.000000	1110.130000	577.405000	468.63
max	19043.138560	1.000000	49039.570000	40761.250000	22500.00

In [51]:
```
df.drop(['CUSTOMER_ID'], axis = 1, inplace= True)
# checking the modified dataset:
df.head()
```

Out[51]:

	BALANCE	BALANCE_FREQ	PURCHASES	ONE_OFF_PURCHASE	INSTALLMENT_PURCHASES
0	40.900749	0.818182	95.40	0.00	95.4
1	3202.467416	0.909091	0.00	0.00	0.0
2	2495.148862	1.000000	773.17	773.17	0.0
3	1666.670542	0.636364	1499.00	1499.00	0.0
4	817.714335	1.000000	16.00	16.00	0.0

In [52]:
```
df.isnull().sum()
```

Out[52]:

```
BALANCE                             0
BALANCE_FREQ                        0
PURCHASES                           0
ONE_OFF_PURCHASE                    0
INSTALLMENT_PURCHASES               0
CASH_IN_ADVANCE                     0
PURCHASE_FREQ                       0
ONE_OFF_PURCHASES_FREQUENCY         0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

In [53]:
```
df ['MINIMUM_PAYMENTS'] = df ['MINIMUM_PAYMENTS'].fillna (0)
df ['CREDIT_LIMIT'] = df ['CREDIT_LIMIT'].fillna (0)
```

In [54]:
```
df.isnull().sum()
```

Out[54]:
```
BALANCE                             0
BALANCE_FREQ                        0
PURCHASES                           0
ONE_OFF_PURCHASE                    0
INSTALLMENT_PURCHASES               0
CASH_IN_ADVANCE                     0
PURCHASE_FREQ                       0
ONE_OFF_PURCHASES_FREQUENCY         0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        0
PAYMENTS                            0
MINIMUM_PAYMENTS                    0
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

In [26]:
```
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.manifold import TSNE
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans # Class to develop kmeans model
from sklearn import metrics
from sklearn.metrics import silhouette_score # base for clustering
from yellowbrick.cluster import SilhouetteVisualizer
from sklearn.mixture import GaussianMixture
```

In [10]:
```
!pip install yellowbrick
```

```
Collecting yellowbrick
  Downloading yellowbrick-1.5-py3-none-any.whl (282 kB)
     ------------------------------------ 282.6/282.6 kB 1.9 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn>=1.0.0 in c:\users\ajithkumar.pola
\toshiba\jupiter\lib\site-packages (from yellowbrick) (1.0.2)
Requirement already satisfied: numpy>=1.16.0 in c:\users\ajithkumar.pola\toshib
a\jupiter\lib\site-packages (from yellowbrick) (1.21.5)
Requirement already satisfied: cycler>=0.10.0 in c:\users\ajithkumar.pola\toshi
ba\jupiter\lib\site-packages (from yellowbrick) (0.11.0)
Requirement already satisfied: scipy>=1.0.0 in c:\users\ajithkumar.pola\toshiba
\jupiter\lib\site-packages (from yellowbrick) (1.9.1)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in c:\users\ajithkuma
r.pola\toshiba\jupiter\lib\site-packages (from yellowbrick) (3.5.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\ajithkumar.pola\toshib
a\jupiter\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (9.2.
0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\ajithkumar.pola\tos
hiba\jupiter\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick)
(3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\ajithkumar.pola\to
shiba\jupiter\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick)
(4.25.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\ajithkumar.pola
\toshiba\jupiter\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbric
k) (2.8.2)
Requirement already satisfied: packaging>=20.0 in c:\users\ajithkumar.pola\tosh
iba\jupiter\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2
1.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\ajithkumar.pola\to
shiba\jupiter\lib\site-packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick)
(1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\ajithkumar.pola
\toshiba\jupiter\lib\site-packages (from scikit-learn>=1.0.0->yellowbrick) (2.
2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\ajithkumar.pola\toshiba
\jupiter\lib\site-packages (from scikit-learn>=1.0.0->yellowbrick) (1.1.0)
Requirement already satisfied: six>=1.5 in c:\users\ajithkumar.pola\toshiba\jup
iter\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.0.0,>=2.0.2->y
ellowbrick) (1.16.0)
Installing collected packages: yellowbrick
Successfully installed yellowbrick-1.5
```

In [27]:
```
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.manifold import TSNE
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans # Class to develop kmeans model
from sklearn import metrics
from sklearn.metrics import silhouette_score # base for clustering
from yellowbrick.cluster import SilhouetteVisualizer
from sklearn.mixture import GaussianMixture
```

In [28]:
```
import warnings
warnings.filterwarnings('ignore')
import os
```

In [30]:
```
from sklearn.decomposition import PCA
```

In [31]:
```
#Scaling the data
scaler = StandardScaler()
scaled_df = scaler.fit_transform(df)

# Normalizing the Data
normalized_df = normalize(scaled_df)

# Converting the numpy array into a pandas DataFrame
normalized_df = pd.DataFrame(normalized_df)

# Reducing the dimensions of the data
pca = PCA(n_components = 2)
X_principal = pca.fit_transform(normalized_df)
X_principal = pd.DataFrame(X_principal)
X_principal.columns = ['P1', 'P2']

X_principal.head(2)
```

Out[31]:

	P1	P2
0	-0.490758	-0.679041
1	-0.518463	0.545358

In [32]:
```
gmm = GaussianMixture(n_components = 3)
gmm.fit(X_principal)
```

Out[32]:
```
GaussianMixture(n_components=3)
```

In [45]:
```
plt.scatter(X_principal['P1'], X_principal['P2'],c = GaussianMixture(n_components
plt.show()
```