In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# ignore warnings
import warnings
warnings.filterwarnings(action="ignore")

# clustering
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.mixture import GaussianMixture
from matplotlib import cm
from sklearn.metrics import silhouette_samples, silhouette_score
```

In [3]:
```python
data = pd.read_csv("C:\\Users\\RAHUL KUMAR UPADHYAY\\Downloads\\CC GENERAL.csv")
```

In [4]:
```python
data
```

Out[4]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTAL
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	
...	...	...	...	...	...	
8945	C19186	28.493517	1.000000	291.12	0.00	
8946	C19187	19.183215	1.000000	300.00	0.00	
8947	C19188	23.398673	0.833333	144.40	0.00	
8948	C19189	13.457564	0.833333	0.00	0.00	
8949	C19190	372.708075	0.666667	1093.25	1093.25	

8950 rows × 18 columns

In [5]:
```python
print('Data shape: ' + str(data.shape))
data.head()
```

```
Data shape: (8950, 18)
```

Out[5]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLM
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 [6]:

```
data.describe()
```

Out[6]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENT
count	8950.000000	8950.000000	8950.000000	8950.000000	
mean	1564.474828	0.877271	1003.204834	592.437371	
std	2081.531879	0.236904	2136.634782	1659.887917	
min	0.000000	0.000000	0.000000	0.000000	
25%	128.281915	0.888889	39.635000	0.000000	
50%	873.385231	1.000000	361.280000	38.000000	
75%	2054.140036	1.000000	1110.130000	577.405000	
max	19043.138560	1.000000	49039.570000	40761.250000	

Data Cleaning

In [7]:

```
data.isna().sum()
```

Out[7]:

```
CUST_ID                               0
BALANCE                               0
BALANCE_FREQUENCY                     0
PURCHASES                             0
ONEOFF_PURCHASES                      0
INSTALLMENTS_PURCHASES                0
CASH_ADVANCE                          0
PURCHASES_FREQUENCY                   0
ONEOFF_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 [9]:
```
# impute with median
data.loc[(data['MINIMUM_PAYMENTS'].isnull()==True),'MINIMUM_PAYMENTS'] = data['M
data.loc[(data['CREDIT_LIMIT'].isnull()==True),'CREDIT_LIMIT'] = data['CREDIT_LI
```

In [10]:
```
# double check
data.isna().sum()
```

Out[10]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_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 [14]:
```
from sklearn.preprocessing import StandardScaler

# Drop the 'CUST_ID' column
data = data.drop("CUST_ID", axis=1)

# Normalize values
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

# Convert back to DataFrame to retain column names
data_scaled = pd.DataFrame(data_scaled, columns=data.columns)

data_scaled.shape
```

Out[14]:
```
(8950, 17)
```

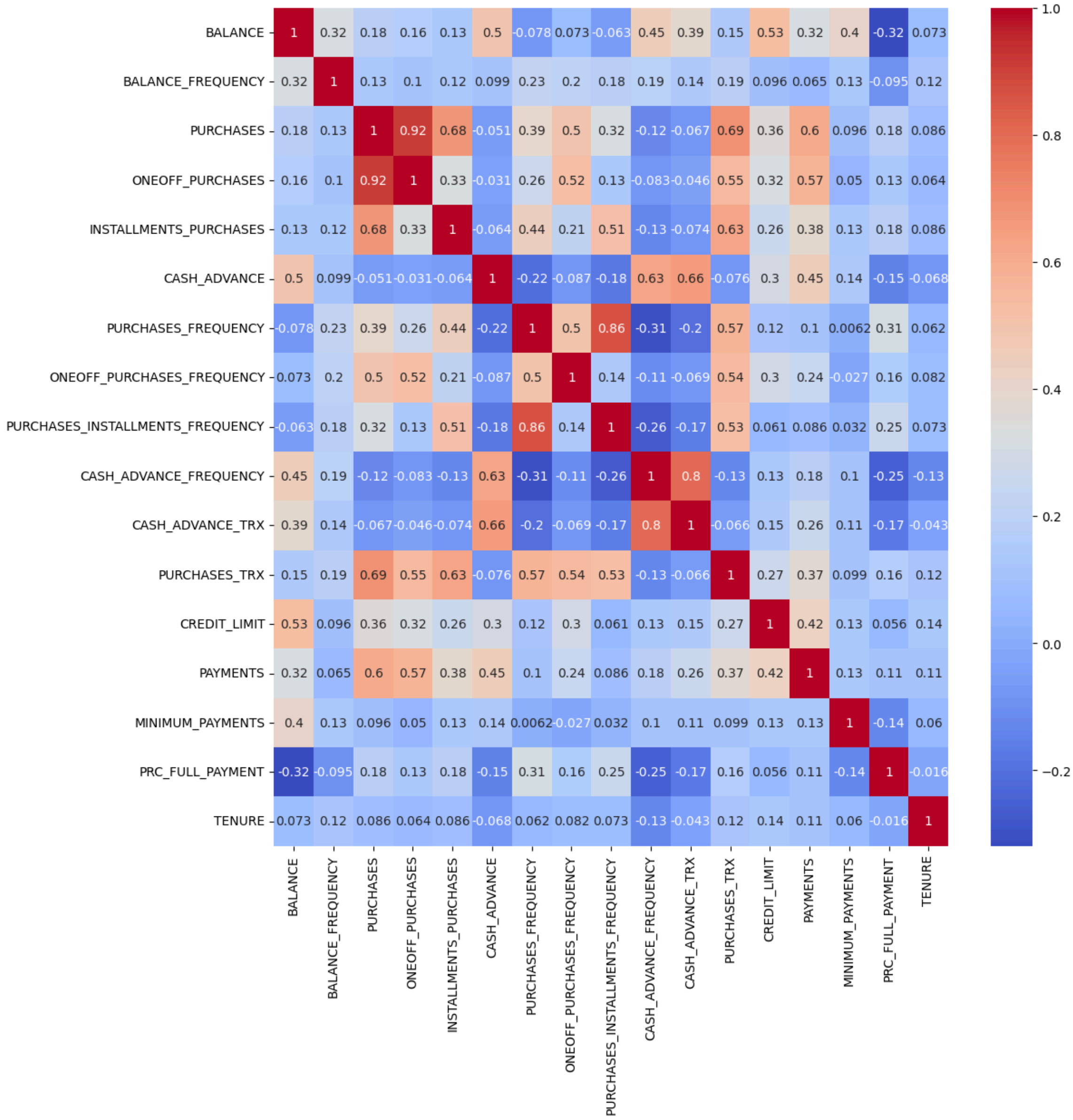
In [15]:
```
data_imputed = pd.DataFrame(data_scaled, columns=data.columns)
```

Clustering

Correlation Check

In [16]:
```
plt.figure(figsize = (12, 12))
sns.heatmap(data_imputed.corr(), annot=True, cmap='coolwarm',
            xticklabels=data_imputed.columns,
            yticklabels=data_imputed.columns)
```

Out[16]:
```
<Axes: >
```

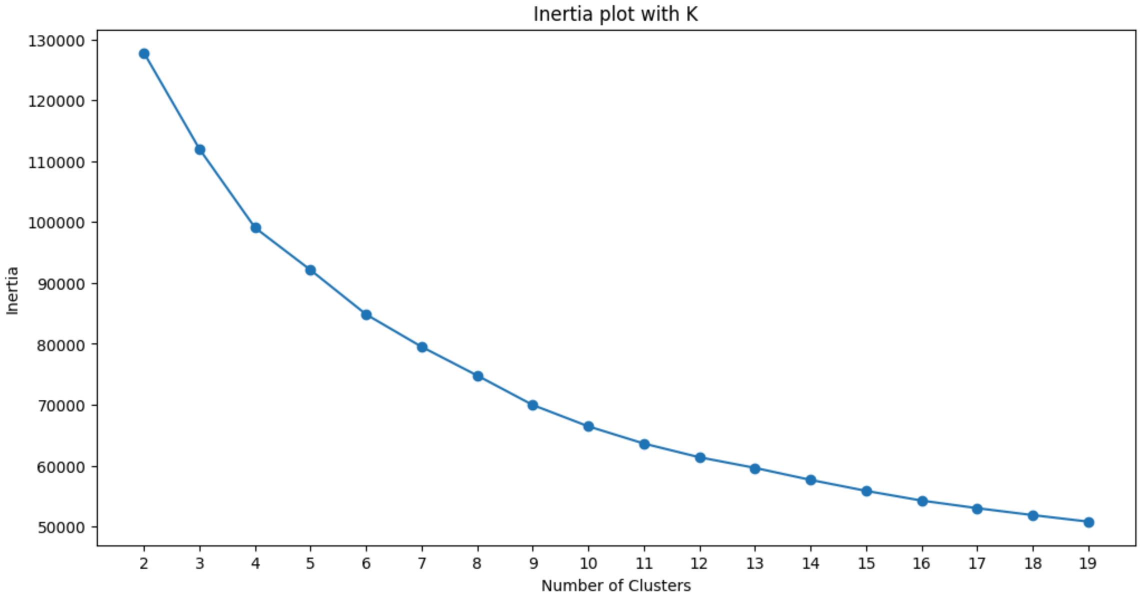


Clustering using K-Means

Inertia Plot

In [17]:
```
# inertia plotter function
def inertia_plot(clust, X, start = 2, stop = 20):
    inertia = []
    for x in range(start,stop):
        km = clust(n_clusters = x)
        labels = km.fit_predict(X)
        inertia.append(km.inertia_)
    plt.figure(figsize = (12,6))
    plt.plot(range(start,stop), inertia, marker = 'o')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia')
    plt.title('Inertia plot with K')
    plt.xticks(list(range(start, stop)))
    plt.show()
```

In [18]:
```
inertia_plot(KMeans, data_imputed)
```



Silhouette Scores

Silhouette analysis can be used to study the separation distance between the resulting clusters. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters and thus provides a way to assess parameters like number of clusters visually. This measure has a range of [-1, 1].

In [19]:
```
def silh_samp_cluster(clust,  X, start=2, stop=5, metric = 'euclidean'):
    # taken from sebastian Raschkas book Python Machine Learning second edition
    for x in range(start, stop):
        km = clust(n_clusters = x)
        y_km = km.fit_predict(X)
        cluster_labels = np.unique(y_km)
        n_clusters = cluster_labels.shape[0]
        silhouette_vals = silhouette_samples(X, y_km, metric = metric)
        y_ax_lower, y_ax_upper =0,0
        yticks = []
        for i, c in enumerate(cluster_labels):
            c_silhouette_vals = silhouette_vals[y_km == c]
            c_silhouette_vals.sort()
            y_ax_upper += len(c_silhouette_vals)
            color = cm.jet(float(i)/n_clusters)
            plt.barh(range(y_ax_lower, y_ax_upper),
                    c_silhouette_vals,
                    height=1.0,
                    edgecolor='none',
                    color = color)
            yticks.append((y_ax_lower + y_ax_upper)/2.)
            y_ax_lower+= len(c_silhouette_vals)

        silhouette_avg = np.mean(silhouette_vals)
        plt.axvline(silhouette_avg,
                   color = 'red',
                   linestyle = "--")
        plt.yticks(yticks, cluster_labels+1)
        plt.ylabel("cluster")
        plt.xlabel('Silhouette Coefficient')
```

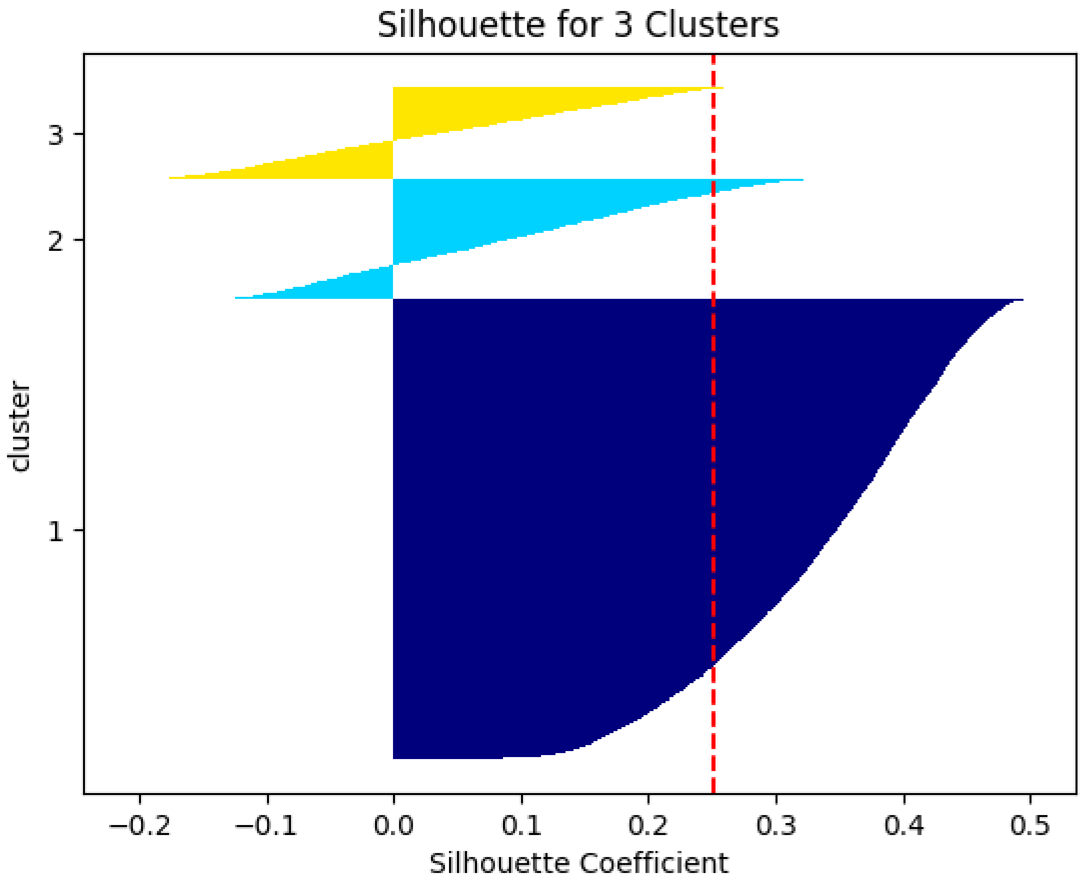
```
        plt.title('Silhouette for ' + str(x) + " Clusters")
        plt.show()
```

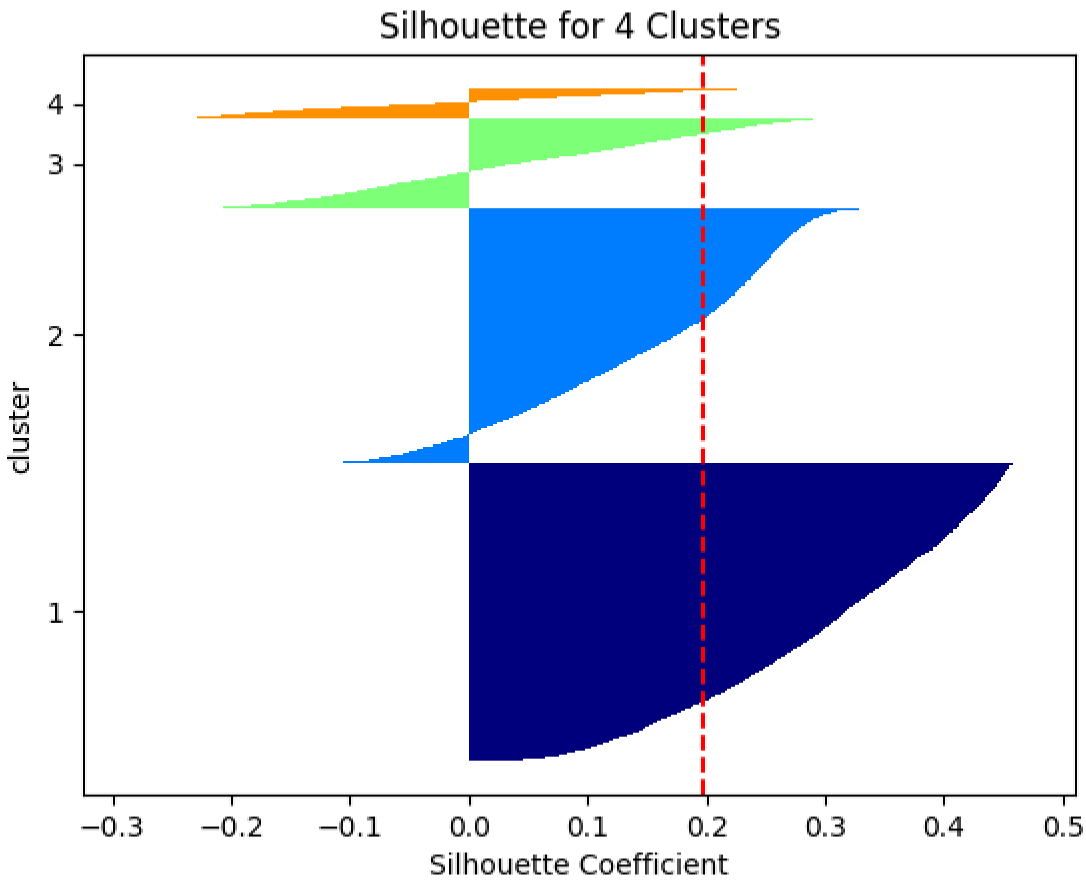
In [20]:
```
for x in range(2, 7):
    alg = KMeans(n_clusters = x, )
    label = alg.fit_predict(data_imputed)
    print('Silhouette-Score for', x,  'Clusters: ', silhouette_score(data_impute
```

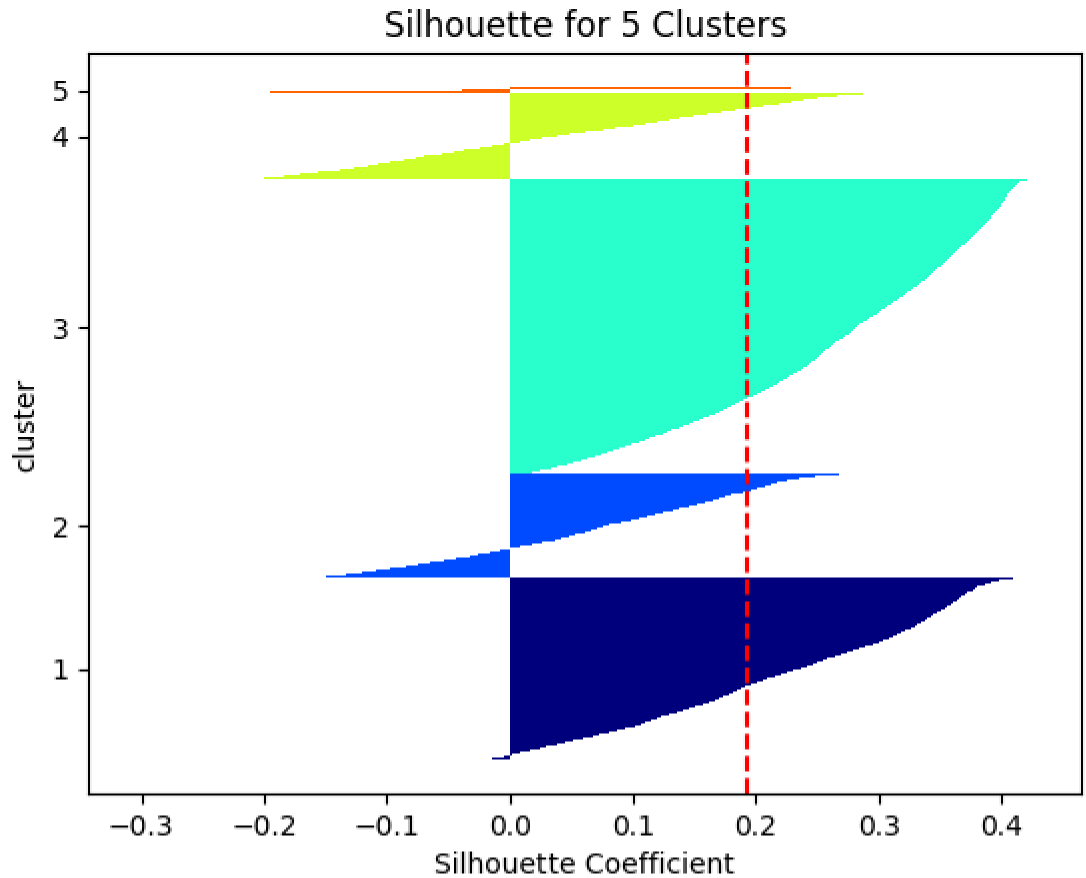
```
Silhouette-Score for 2 Clusters:  0.20974613933248726
Silhouette-Score for 3 Clusters:  0.25061926305697263
Silhouette-Score for 4 Clusters:  0.197301890321782
Silhouette-Score for 5 Clusters:  0.19325195080511473
Silhouette-Score for 6 Clusters:  0.20286011584987834
```

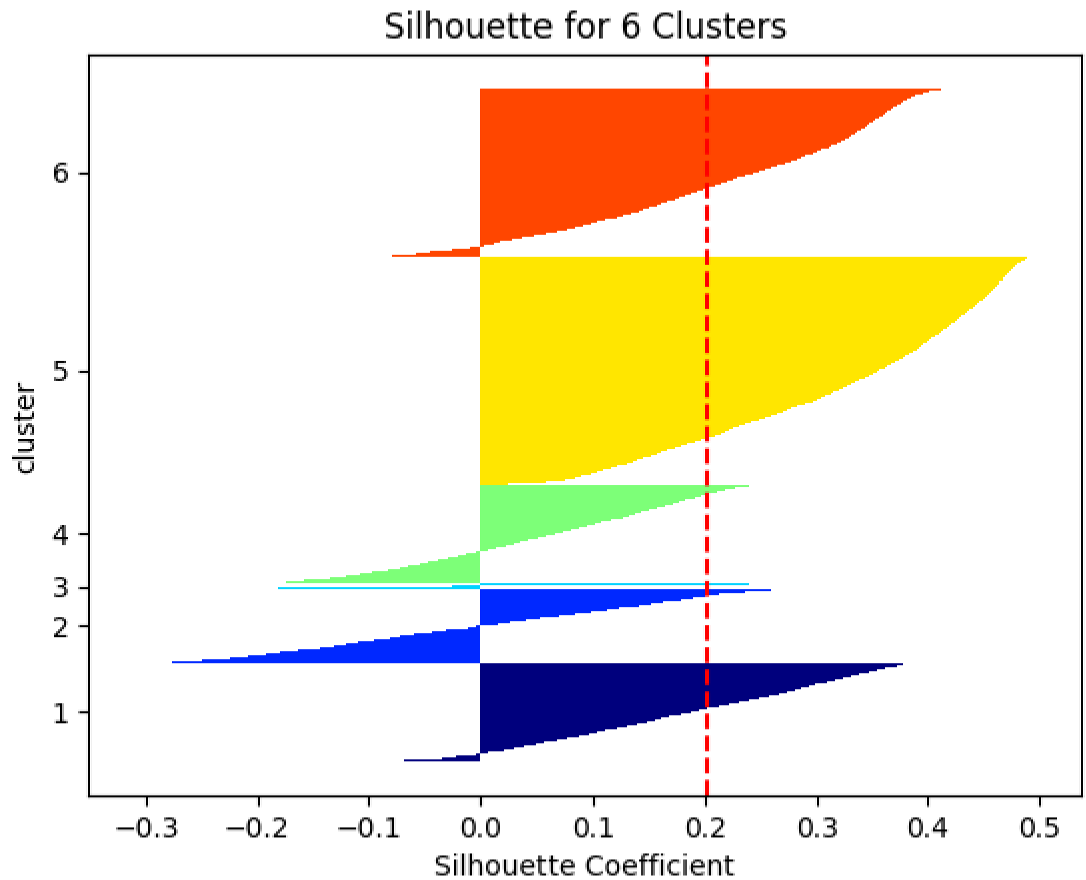
Silhouette plots:

In [21]:
```
silh_samp_cluster(KMeans, data_imputed, stop=7)
```


Silhouette for 3 Clusters
cluster
3
2
1
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
Silhouette Coefficient


Silhouette for 4 Clusters
cluster
4
3
2
1
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
Silhouette Coefficient





Feature Extraction with PCA

Clustering Metrics

In [22]:
```
# apply PCA and display clustering metrics
for y in range(2, 5):
    print("PCA with # of components: ", y)
    pca = PCA(n_components=y)
    data_p = pca.fit_transform(data_imputed)
    for x in range(2, 7):
        alg = KMeans(n_clusters = x, )
        label = alg.fit_predict(data_p)
        print('Silhouette-Score for', x,  'Clusters: ', silhouette_score(data_p,
    print()
```

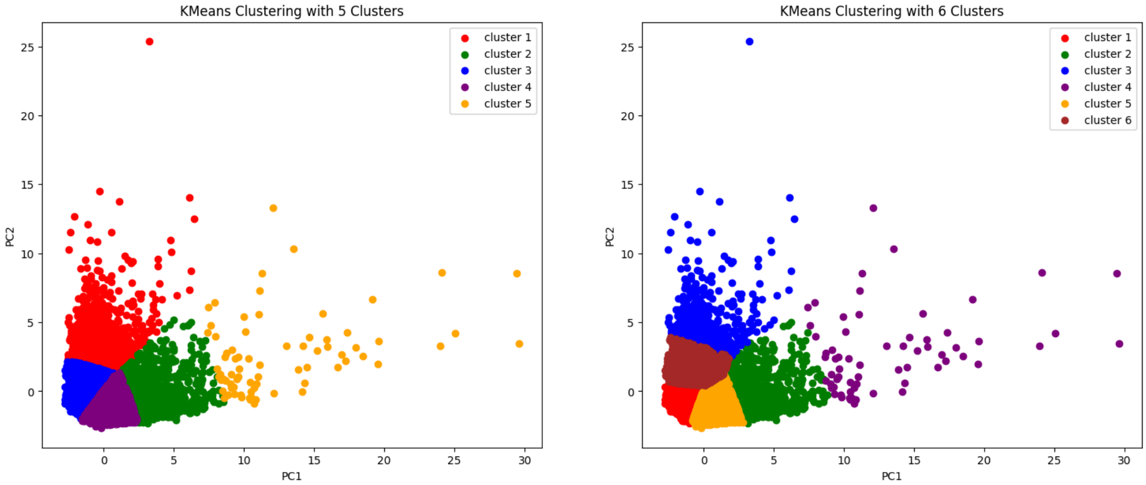
```
PCA with # of components:  2
Silhouette-Score for 2 Clusters:  0.46396737848113945        Inertia:  49682.72
4336380554
Silhouette-Score for 3 Clusters:  0.450649491934593        Inertia:  33030.8335
1306216
Silhouette-Score for 4 Clusters:  0.40734931095229193        Inertia:  24544.31
2345954524
Silhouette-Score for 5 Clusters:  0.40110663555137965        Inertia:  19475.98
0412888395
Silhouette-Score for 6 Clusters:  0.38857410945086657        Inertia:  16231.27
0377093988

PCA with # of components:  3
Silhouette-Score for 2 Clusters:  0.34125043193355736        Inertia:  62045.71
043206413
Silhouette-Score for 3 Clusters:  0.3800481096977205        Inertia:  46325.304
16733775
Silhouette-Score for 4 Clusters:  0.36913871000448595        Inertia:  34659.81
458535303
Silhouette-Score for 5 Clusters:  0.36822773815403004        Inertia:  28591.74
3243414243
Silhouette-Score for 6 Clusters:  0.33071571513271814        Inertia:  24848.42
0007445857

PCA with # of components:  4
Silhouette-Score for 2 Clusters:  0.305683674794164        Inertia:  73185.0062
5624276
Silhouette-Score for 3 Clusters:  0.3430650328703002        Inertia:  57561.137
46340335
Silhouette-Score for 4 Clusters:  0.3219170281283231        Inertia:  45288.097
873435865
Silhouette-Score for 5 Clusters:  0.3200740225469506        Inertia:  39166.423
30583867
Silhouette-Score for 6 Clusters:  0.28859973478461914        Inertia:  35301.64
947994424
```

Visualization

In [23]:
```
data_p = pd.DataFrame(PCA(n_components = 2).fit_transform(data_imputed))
preds = pd.Series(KMeans(n_clusters = 5,).fit_predict(data_p))
data_p = pd.concat([data_p, preds], axis =1)
data_p.columns = [0,1,'target']
```

```
fig = plt.figure(figsize = (18, 7))
colors = ['red', 'green', 'blue', 'purple', 'orange', 'brown']
plt.subplot(121)
plt.scatter(data_p[data_p['target']==0].iloc[:,0], data_p[data_p.target==0].iloc
plt.scatter(data_p[data_p['target']==1].iloc[:,0], data_p[data_p.target==1].iloc
plt.scatter(data_p[data_p['target']==2].iloc[:,0], data_p[data_p.target==2].iloc
plt.scatter(data_p[data_p['target']==3].iloc[:,0], data_p[data_p.target==3].iloc
plt.scatter(data_p[data_p['target']==4].iloc[:,0], data_p[data_p.target==4].iloc
plt.legend()
plt.title('KMeans Clustering with 5 Clusters')
plt.xlabel('PC1')
plt.ylabel('PC2')


data_p = pd.DataFrame(PCA(n_components = 2).fit_transform(data_imputed))
preds = pd.Series(KMeans(n_clusters = 6,).fit_predict(data_p))
data_p = pd.concat([data_p, preds], axis =1)
data_p.columns = [0,1,'target']

plt.subplot(122)
plt.scatter(data_p[data_p['target']==0].iloc[:,0], data_p[data_p.target==0].iloc
plt.scatter(data_p[data_p['target']==1].iloc[:,0], data_p[data_p.target==1].iloc
plt.scatter(data_p[data_p['target']==2].iloc[:,0], data_p[data_p.target==2].iloc
plt.scatter(data_p[data_p['target']==3].iloc[:,0], data_p[data_p.target==3].iloc
plt.scatter(data_p[data_p['target']==4].iloc[:,0], data_p[data_p.target==4].iloc
plt.scatter(data_p[data_p['target']==5].iloc[:,0], data_p[data_p.target==5].iloc
plt.legend()
plt.title('KMeans Clustering with 6 Clusters')
plt.xlabel('PC1')
plt.ylabel('PC2')
```

Out[23]: `Text(0, 0.5, 'PC2')`



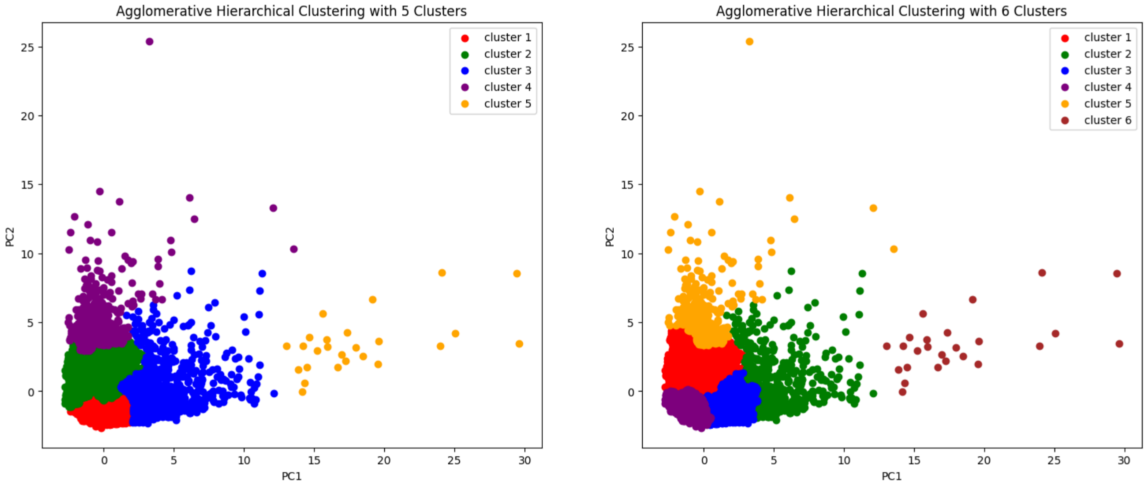
Agglomerative Hierarchical Clustering with PCA

In [24]:

```
data_p = pd.DataFrame(PCA(n_components = 2).fit_transform(data_imputed))
preds = pd.Series(AgglomerativeClustering(n_clusters = 5,).fit_predict(data_p))
data_p = pd.concat([data_p, preds], axis =1)
data_p.columns = [0,1,'target']

fig = plt.figure(figsize = (18, 7))
```

```
colors = ['red', 'green', 'blue', 'purple', 'orange', 'brown']
plt.subplot(121)
plt.scatter(data_p[data_p['target']==0].iloc[:,0], data_p[data_p.target==0].iloc
plt.scatter(data_p[data_p['target']==1].iloc[:,0], data_p[data_p.target==1].iloc
plt.scatter(data_p[data_p['target']==2].iloc[:,0], data_p[data_p.target==2].iloc
plt.scatter(data_p[data_p['target']==3].iloc[:,0], data_p[data_p.target==3].iloc
plt.scatter(data_p[data_p['target']==4].iloc[:,0], data_p[data_p.target==4].iloc
plt.legend()
plt.title('Agglomerative Hierarchical Clustering with 5 Clusters')
plt.xlabel('PC1')
plt.ylabel('PC2')


data_p = pd.DataFrame(PCA(n_components = 2).fit_transform(data_imputed))
preds = pd.Series(AgglomerativeClustering(n_clusters = 6,).fit_predict(data_p))
data_p = pd.concat([data_p, preds], axis =1)
data_p.columns = [0,1,'target']

plt.subplot(122)
plt.scatter(data_p[data_p['target']==0].iloc[:,0], data_p[data_p.target==0].iloc
plt.scatter(data_p[data_p['target']==1].iloc[:,0], data_p[data_p.target==1].iloc
plt.scatter(data_p[data_p['target']==2].iloc[:,0], data_p[data_p.target==2].iloc
plt.scatter(data_p[data_p['target']==3].iloc[:,0], data_p[data_p.target==3].iloc
plt.scatter(data_p[data_p['target']==4].iloc[:,0], data_p[data_p.target==4].iloc
plt.scatter(data_p[data_p['target']==5].iloc[:,0], data_p[data_p.target==5].iloc
plt.legend()
plt.title('Agglomerative Hierarchical Clustering with 6 Clusters')
plt.xlabel('PC1')
plt.ylabel('PC2')
```

Out[24]: Text(0, 0.5, 'PC2')



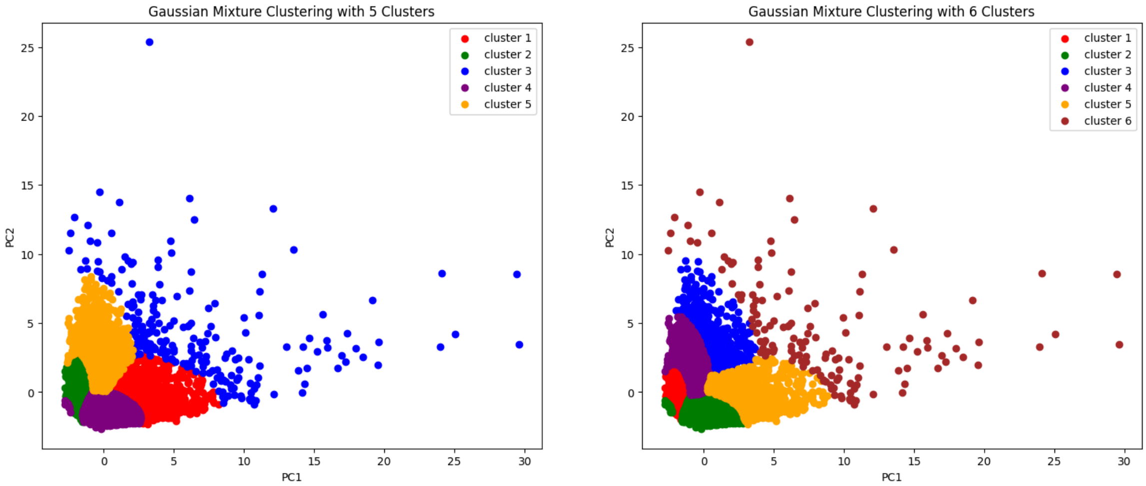
Gaussian Mixture Clustering with PCA

In [25]:

```
data_p = pd.DataFrame(PCA(n_components = 2).fit_transform(data_imputed))
preds = pd.Series(GaussianMixture(n_components = 5,).fit_predict(data_p))
data_p = pd.concat([data_p, preds], axis =1)
data_p.columns = [0,1,'target']

fig = plt.figure(figsize = (18, 7))
colors = ['red', 'green', 'blue', 'purple', 'orange', 'brown']
plt.subplot(121)
plt.scatter(data_p[data_p['target']==0].iloc[:,0], data_p[data_p.target==0].iloc
```

```
plt.scatter(data_p[data_p['target']==1].iloc[:,0], data_p[data_p.target==1].iloc
plt.scatter(data_p[data_p['target']==2].iloc[:,0], data_p[data_p.target==2].iloc
plt.scatter(data_p[data_p['target']==3].iloc[:,0], data_p[data_p.target==3].iloc
plt.scatter(data_p[data_p['target']==4].iloc[:,0], data_p[data_p.target==4].iloc
plt.legend()
plt.title('Gaussian Mixture Clustering with 5 Clusters')
plt.xlabel('PC1')
plt.ylabel('PC2')


data_p = pd.DataFrame(PCA(n_components = 2).fit_transform(data_imputed))
preds = pd.Series(GaussianMixture(n_components = 6,).fit_predict(data_p))
data_p = pd.concat([data_p, preds], axis =1)
data_p.columns = [0,1,'target']

plt.subplot(122)
plt.scatter(data_p[data_p['target']==0].iloc[:,0], data_p[data_p.target==0].iloc
plt.scatter(data_p[data_p['target']==1].iloc[:,0], data_p[data_p.target==1].iloc
plt.scatter(data_p[data_p['target']==2].iloc[:,0], data_p[data_p.target==2].iloc
plt.scatter(data_p[data_p['target']==3].iloc[:,0], data_p[data_p.target==3].iloc
plt.scatter(data_p[data_p['target']==4].iloc[:,0], data_p[data_p.target==4].iloc
plt.scatter(data_p[data_p['target']==5].iloc[:,0], data_p[data_p.target==5].iloc
plt.legend()
plt.title('Gaussian Mixture Clustering with 6 Clusters')
plt.xlabel('PC1')
plt.ylabel('PC2')
```

Out[25]: Text(0, 0.5, 'PC2')



Exploratory Data Analysis

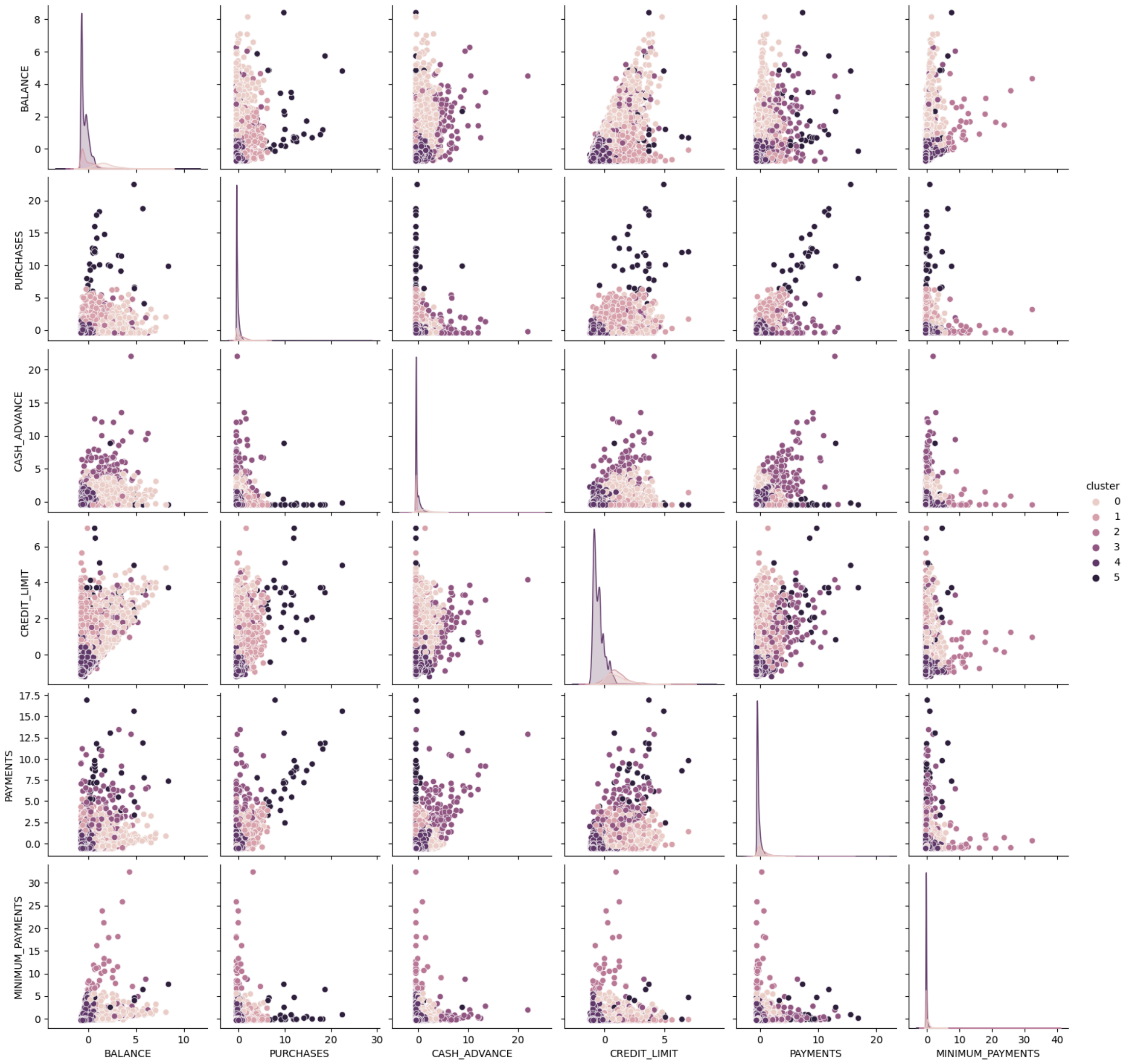
In [26]:
```
# select best columns
best_cols = ["BALANCE", "PURCHASES", "CASH_ADVANCE", "CREDIT_LIMIT", "PAYMENTS",

# dataframe with best columns
data_final = pd.DataFrame(data_imputed[best_cols])

print('New dataframe with best columns has just been created. Data shape: ' + st
```

```
New dataframe with best columns has just been created. Data shape: (8950, 6)
```

In [27]:
```
# apply KMeans clustering
alg = KMeans(n_clusters = 6)
```

```
label = alg.fit_predict(data_final)

# create a 'cluster' column
data_final['cluster'] = label
best_cols.append('cluster')

# make a Seaborn pairplot
sns.pairplot(data_final[best_cols], hue='cluster')
```

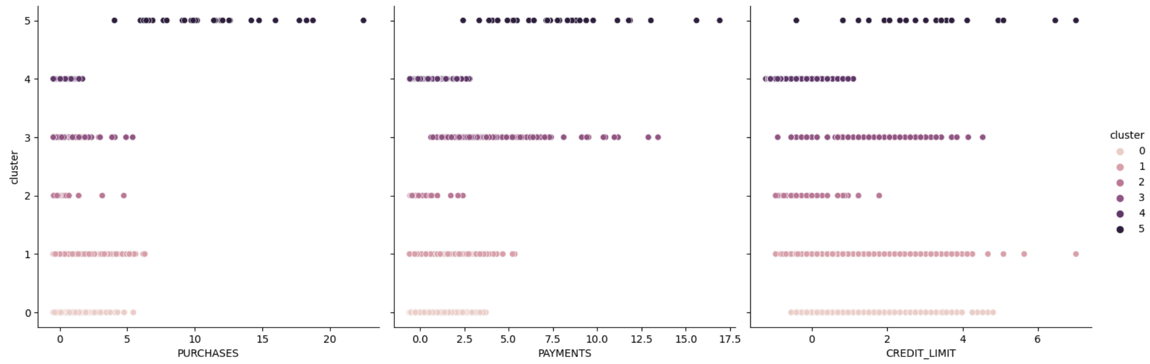
Out[27]: <seaborn.axisgrid.PairGrid at 0x250f9086550>



Cluster 0 (Blue): The Average Joe

In [28]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES', 'PAYMENT
             y_vars=['cluster'],
             height=5, aspect=1)
```

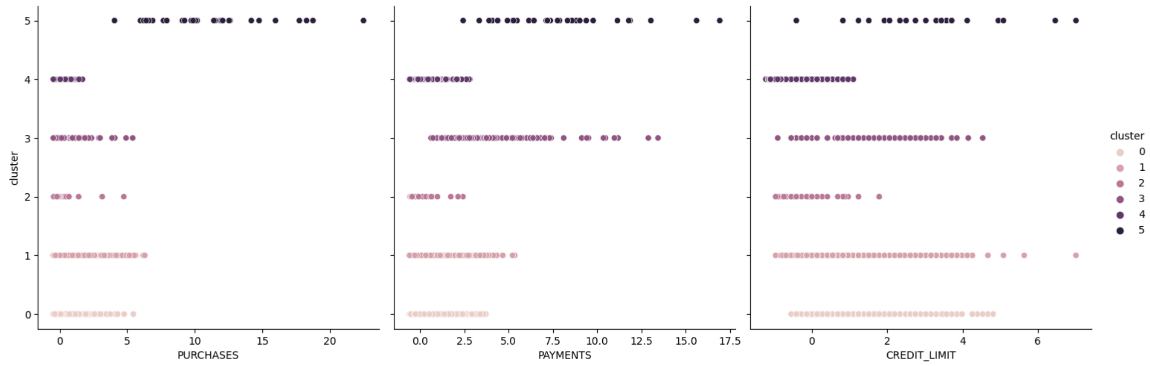
Out[28]: <seaborn.axisgrid.PairGrid at 0x250f7f6d490>



Cluster 1 (Orange): The Active Users

In [29]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES', 'PAYMENT
             y_vars=['cluster'],
             height=5, aspect=1)
```

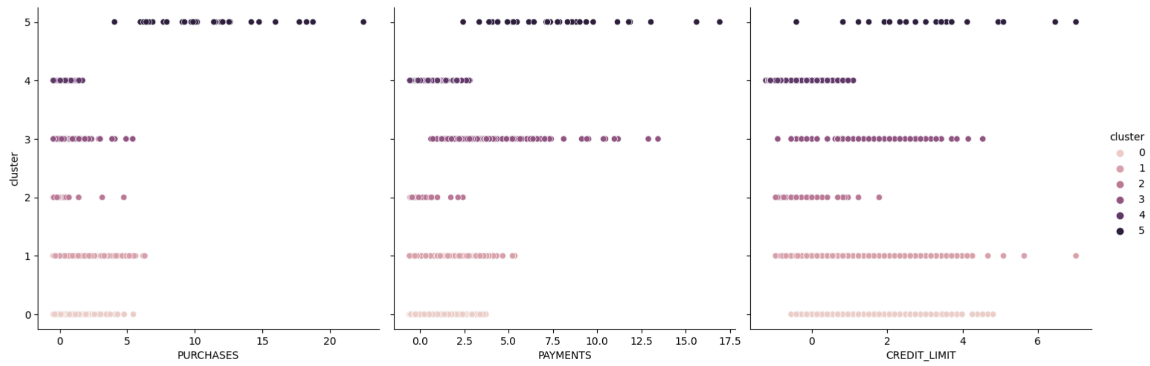
Out[29]: <seaborn.axisgrid.PairGrid at 0x250fb838cd0>



Cluster 2 (Green): The Big Spenders

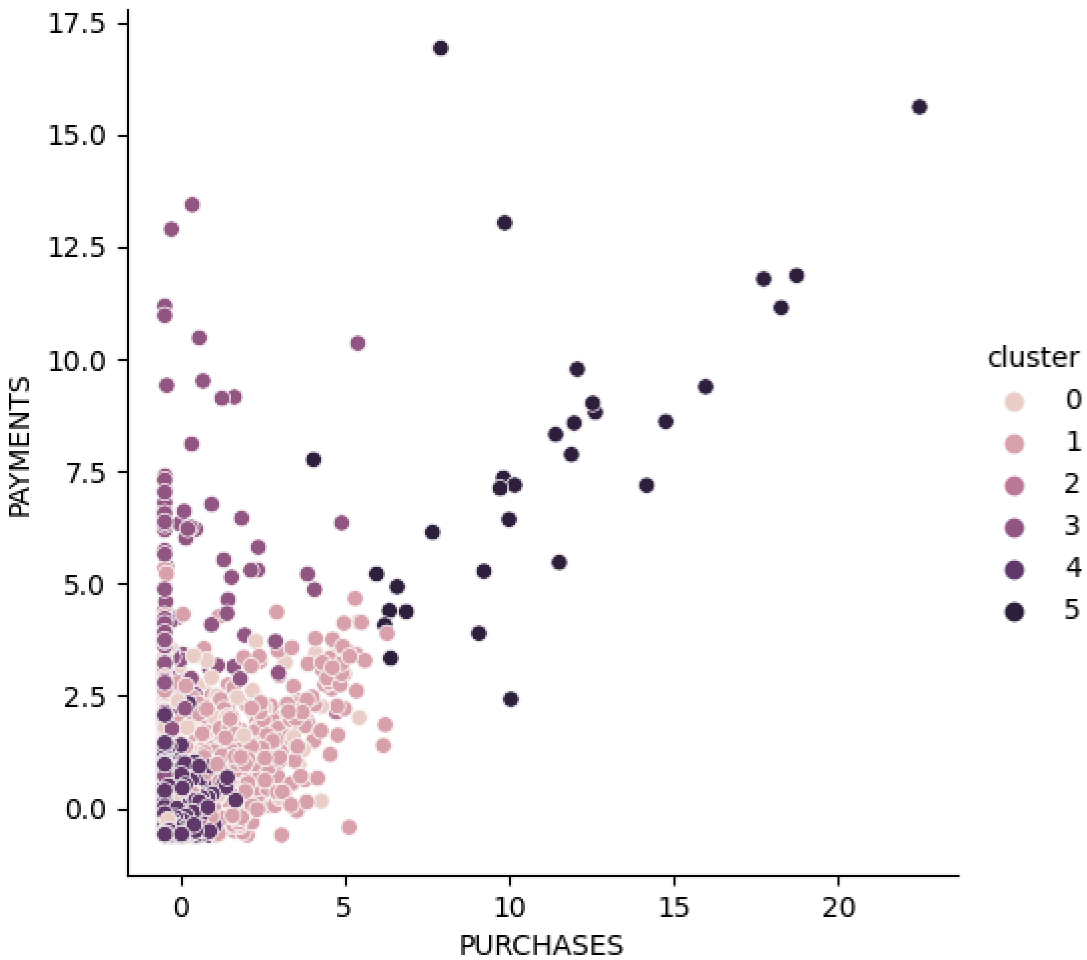
In [30]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES', 'PAYMENT
             height=5, aspect=1)
```

Out[30]: <seaborn.axisgrid.PairGrid at 0x250fa8619d0>



In [31]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES'], y_vars=
             height=5, aspect=1)
```

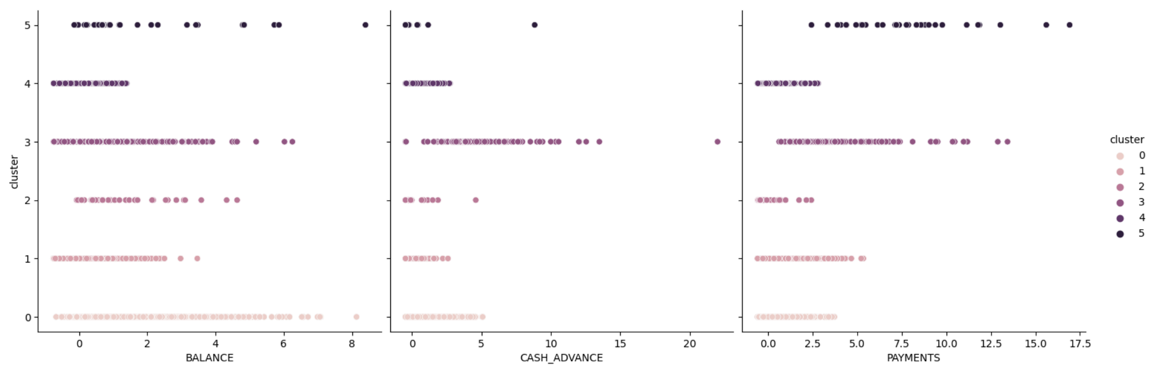
Out[31]: <seaborn.axisgrid.PairGrid at 0x250f7a51350>



Cluster 3 (Red): The Money Borrowers

In [32]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['BALANCE', 'CASH_ADVA
            y_vars=['cluster'],
            height=5, aspect=1)
```

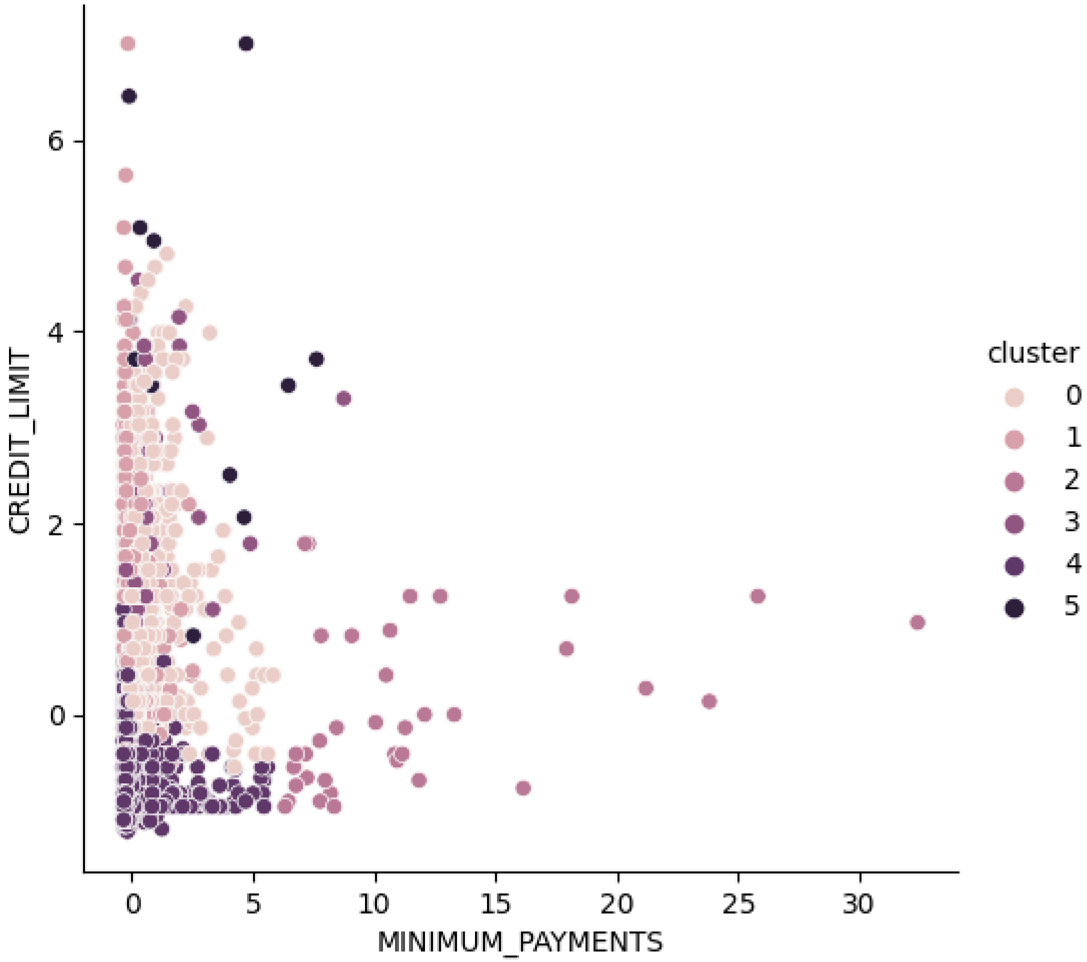
Out[32]: <seaborn.axisgrid.PairGrid at 0x250fbe0d950>



Cluster 4 (Purple): The High Riskers

In [33]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['MINIMUM_PAYMENTS'],
            height=5, aspect=1)
```

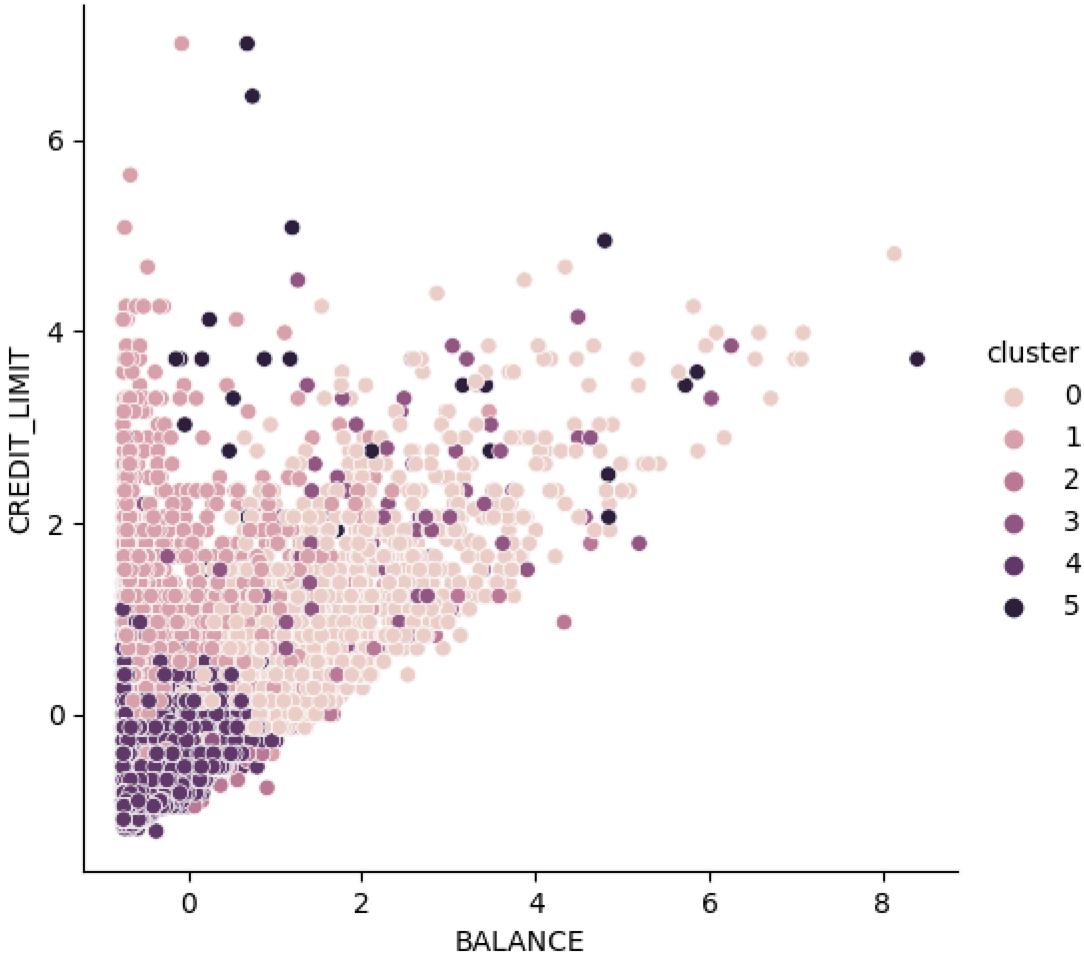
Out[33]: <seaborn.axisgrid.PairGrid at 0x250f7320110>



Cluster 5 (Brown): The Wildcards

In [34]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['BALANCE'], y_vars=['
             height=5, aspect=1)
```

Out[34]: `<seaborn.axisgrid.PairGrid at 0x250fbf4b350>`



Summary and Possible Marketing Strategy

We have learned a lot from this dataset by segmenting the customers into six smaller groups: the Average Joe, the Active Users, the Big Spenders, the Money Borrowers, the High Riskers, and the Wildcards. To conclude this cluster analysis, let's sum up what we have learned and some possible marketing strategies:

The Average Joe do not use credit card very much in their daily life. They have healthy finances and low debts. While encouraging these people to use credit cards more is necessary for the company's profit, business ethics and social responsibility should also be considered.

Identify active customers in order to apply proper marketing strategy towards them. These people are the main group that we should focus on.

Some people are just bad at finance management - for example, the Money Borrowers. This should not be taken lightly.

Although we are currently doing a good job at managing the High Riskers by giving them low credit limits, more marketing strategies targeting this group of customers should be considered.

In [ ]:

```

```

In [ ]:

```

```