Credit Card Customer Segmentation

In [1]:

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

Load the Data

In [3]:

```
df = pd.read_csv('CC GENERAL.csv')
```

In [4]:

```
df.head()
```

Out[4]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_
0	C10001	40.900749	0.818182	95.40	0.00	95.4	
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	64
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	2
4	C10005	817.714335	1.000000	16.00	16.00	0.0	

In [5]:

```
df.describe()
```

Out[5]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVA
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.00
mean	1564.474828	0.877271	1003.204834	592.437371	411.067645	978.87
std	2081.531879	0.236904	2136.634782	1659.887917	904.338115	2097.16
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	128.281915	0.888889	39.635000	0.000000	0.000000	0.00
50%	873.385231	1.000000	361.280000	38.000000	89.000000	0.00
75%	2054.140036	1.000000	1110.130000	577.405000	468.637500	1113.82
max	19043.138560	1.000000	49039.570000	40761.250000	22500.000000	47137.21

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   CUST_ID                           8950 non-null   object
```

```
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_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 [7]:

```
df.isna().mean()*100
```

Out[7]:

```
CUST_ID                             0.000000
BALANCE                             0.000000
BALANCE_FREQUENCY                   0.000000
PURCHASES                           0.000000
ONEOFF_PURCHASES                    0.000000
INSTALLMENTS_PURCHASES              0.000000
CASH_ADVANCE                        0.000000
PURCHASES_FREQUENCY                 0.000000
ONEOFF_PURCHASES_FREQUENCY          0.000000
PURCHASES_INSTALLMENTS_FREQUENCY    0.000000
CASH_ADVANCE_FREQUENCY              0.000000
CASH_ADVANCE_TRX                    0.000000
PURCHASES_TRX                       0.000000
CREDIT_LIMIT                        0.011173
PAYMENTS                            0.000000
MINIMUM_PAYMENTS                    3.497207
PRC_FULL_PAYMENT                    0.000000
TENURE                              0.000000
dtype: float64
```

Data Processing

In [8]:

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

In [9]:

```
df.dropna(subset=['CREDIT_LIMIT'], inplace=True)
```

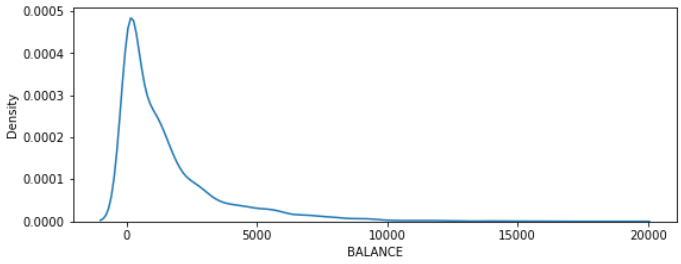
In [10]:

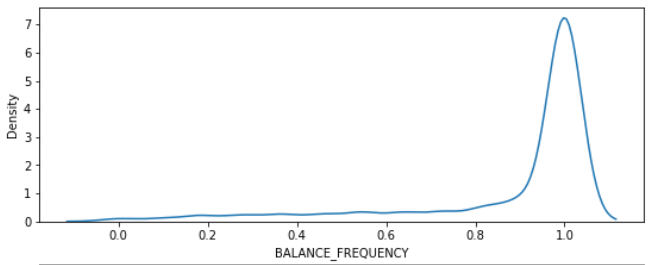
```
df['MINIMUM_PAYMENTS'].fillna(df['MINIMUM_PAYMENTS'].median(), inplace=True)
```

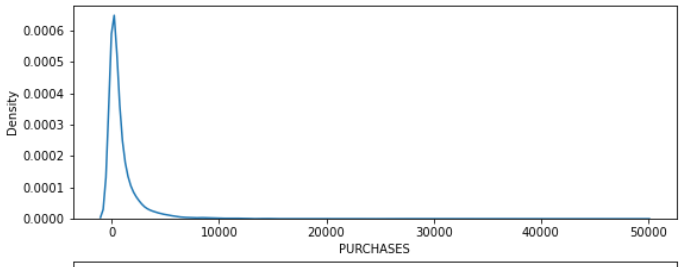
In [11]:

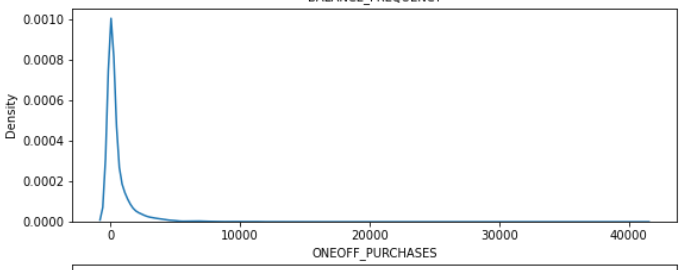
```
plt.figure(figsize=(20,35))
for i, col in enumerate(df.columns):
    if df[col].dtype != 'object':
        ax = plt.subplot(9, 2, i+1)
        sns.kdeplot(df[col], ax=ax)
        plt.xlabel(col)
```

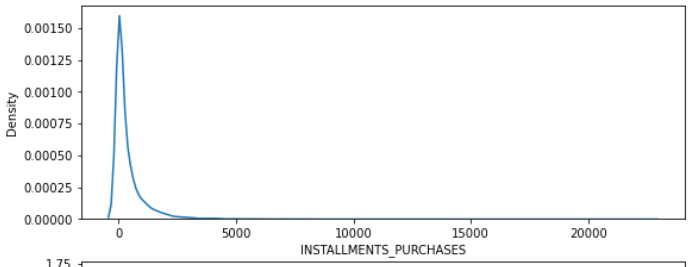
```
plt.show()
```

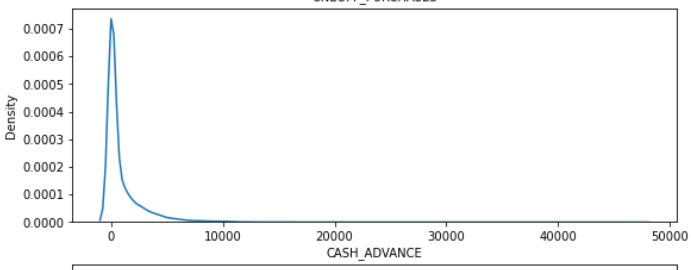

Density
BALANCE

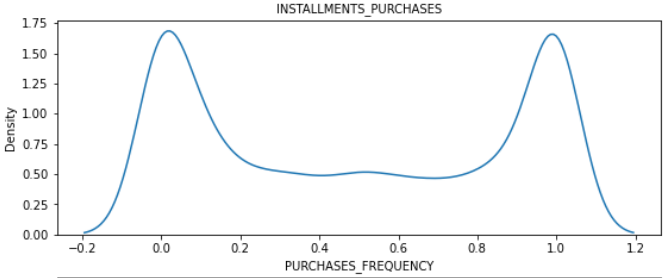

Density
BALANCE_FREQUENCY

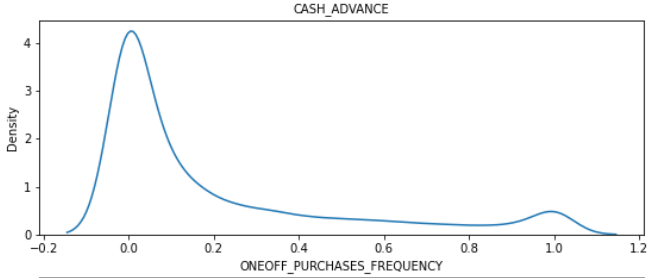

Density
PURCHASES

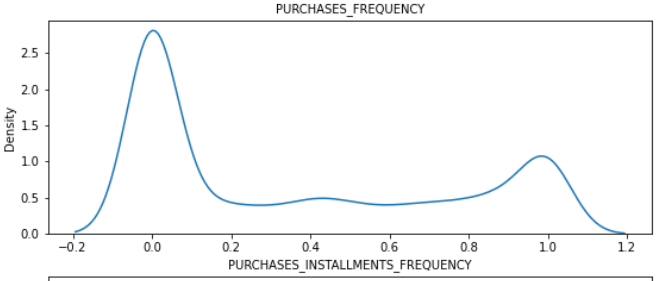

Density
ONEOFF_PURCHASES

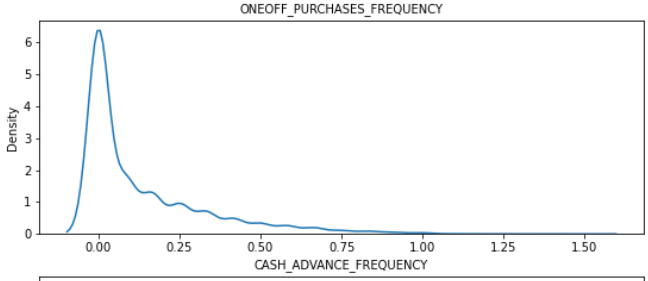

Density
INSTALLMENTS_PURCHASES

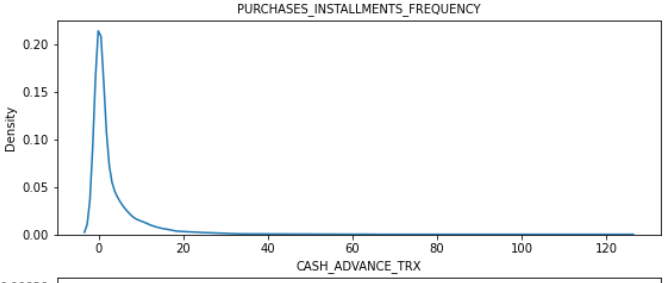

Density
CASH_ADVANCE

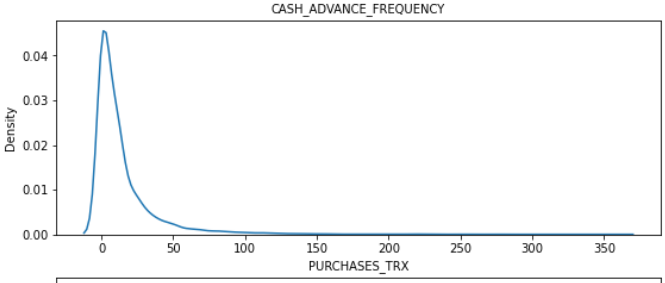

Density
PURCHASES_FREQUENCY

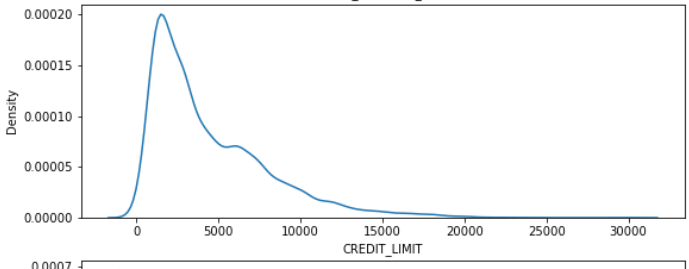

Density
ONEOFF_PURCHASES_FREQUENCY

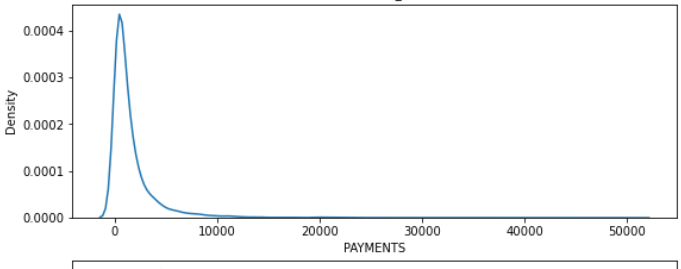

Density
PURCHASES_INSTALLMENTS_FREQUENCY

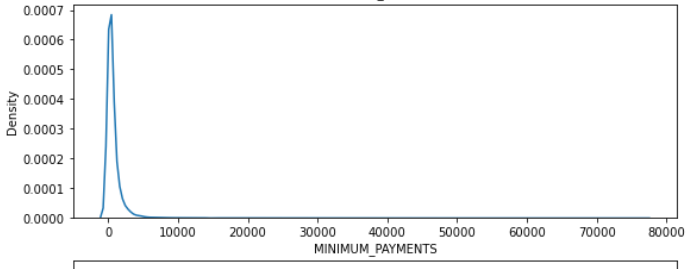

Density
CASH_ADVANCE_FREQUENCY

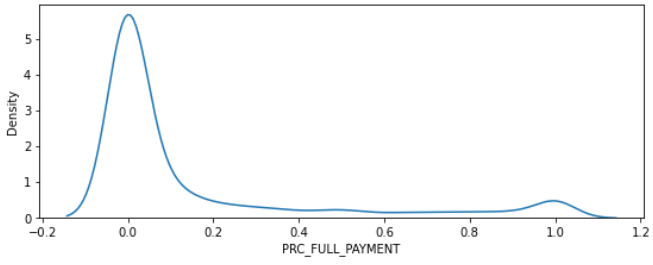

Density
CASH_ADVANCE_TRX

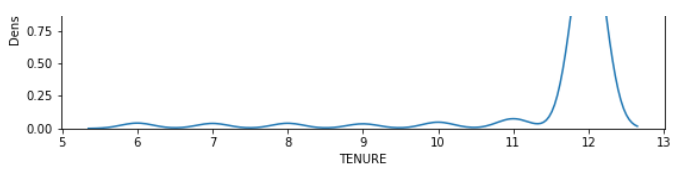

Density
PURCHASES_TRX


Density
CREDIT_LIMIT


Density
PAYMENTS


Density
MINIMUM_PAYMENTS


Density
PRC_FULL_PAYMENT



In [12]:

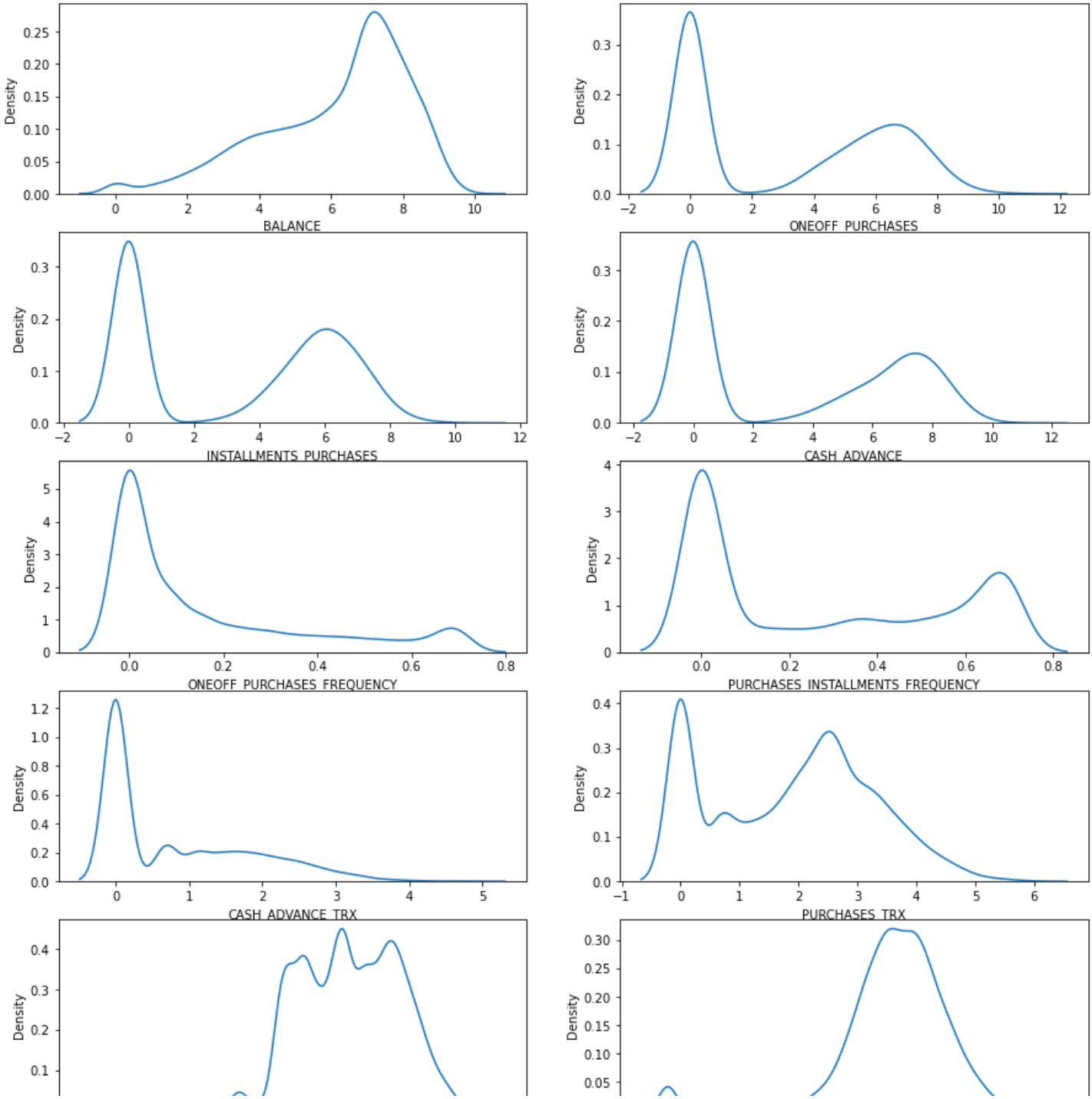
```
cols = ['BALANCE', 'ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE', 'ONEOFF_
PURCHASES_FREQUENCY','PURCHASES_INSTALLMENTS_FREQUENCY', 'CASH_ADVANCE_TRX', 'PURCHASES_T
RX', 'CREDIT_LIMIT', 'PAYMENTS', 'MINIMUM_PAYMENTS', 'PRC_FULL_PAYMENT']
```

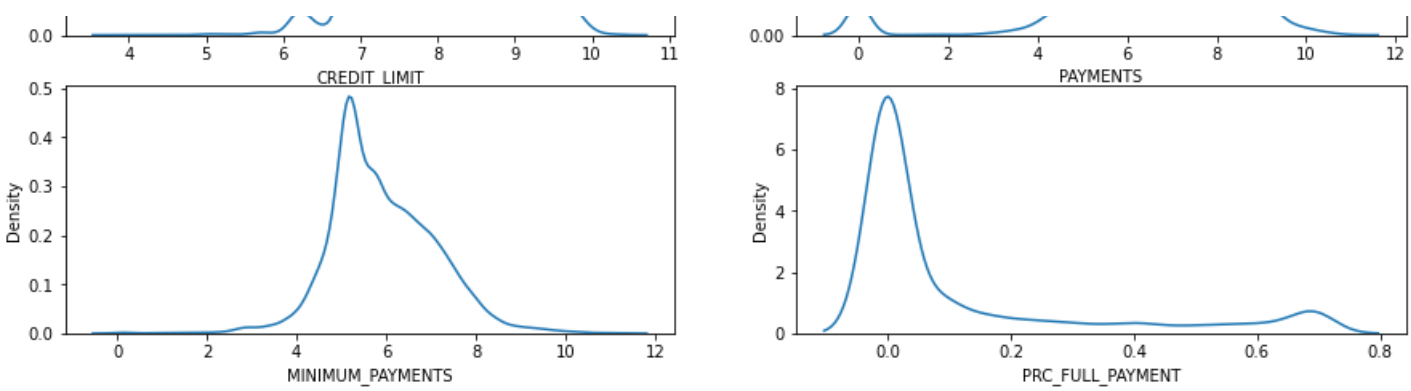
In [13]:

```
for col in cols:
    df[col] = np.log(1 + df[col])
```

In [14]:

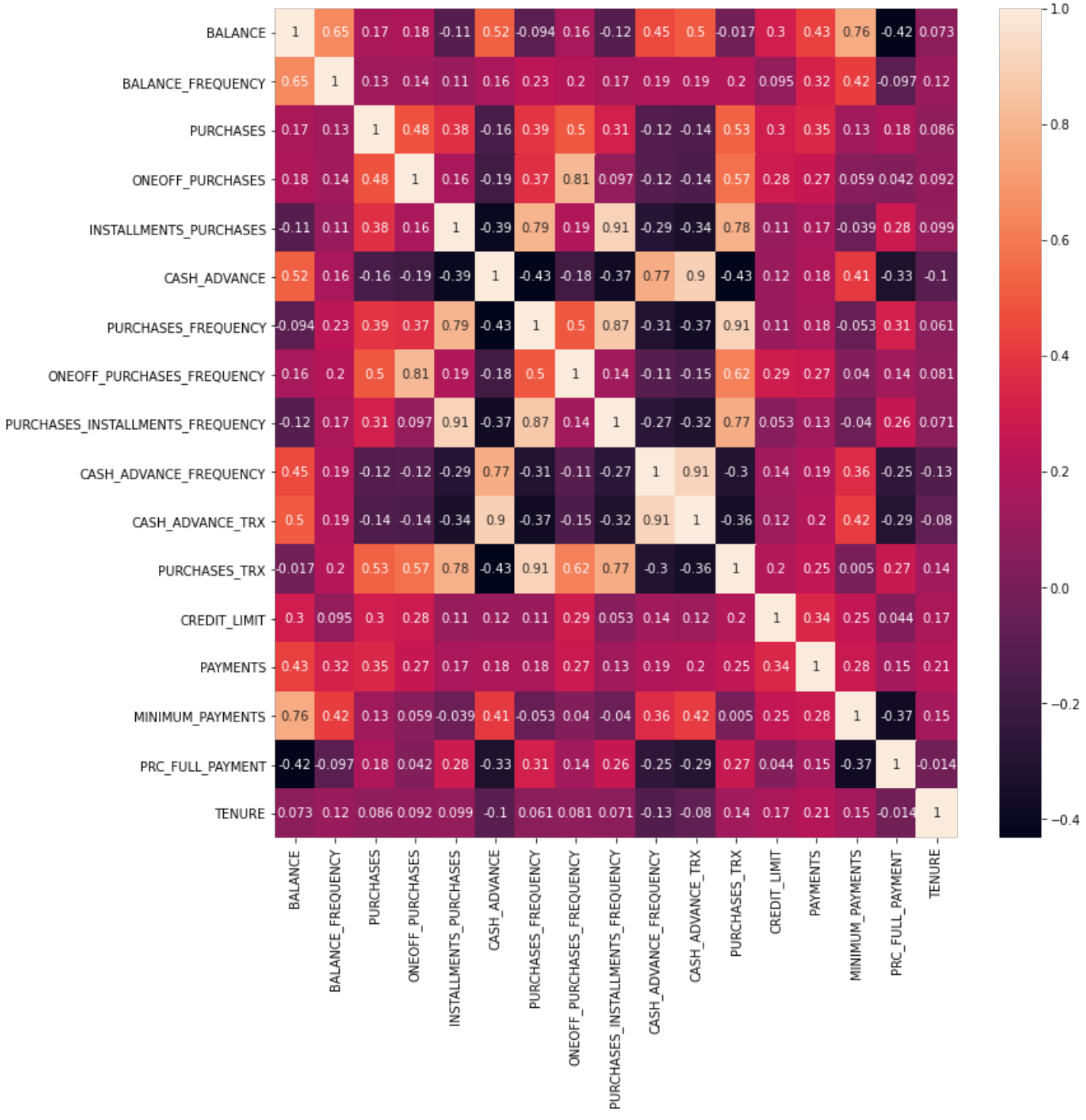
```
plt.figure(figsize=(15,20))
for i, col in enumerate(cols):
    ax = plt.subplot(6, 2, i+1)
    sns.kdeplot(df[col], ax=ax)
plt.show()
```





In [15]:

```
plt.figure(figsize=(12,12))
sns.heatmap(df.corr(), annot=True)
plt.show()
```



In [16]:

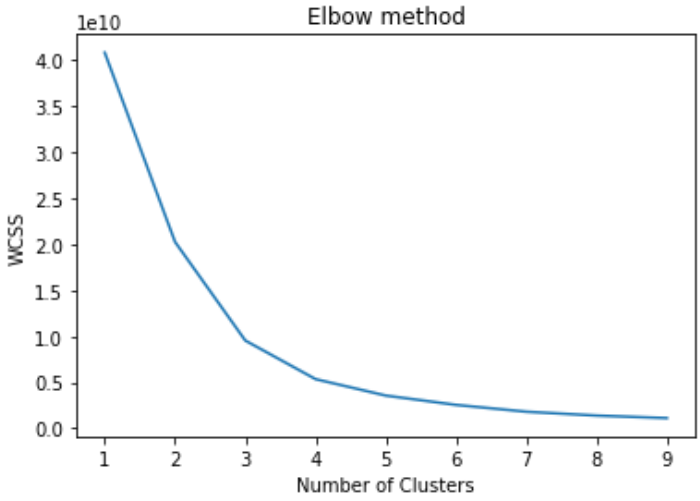
```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=0.95)
X_red = pca.fit_transform(df)
```

Model Training

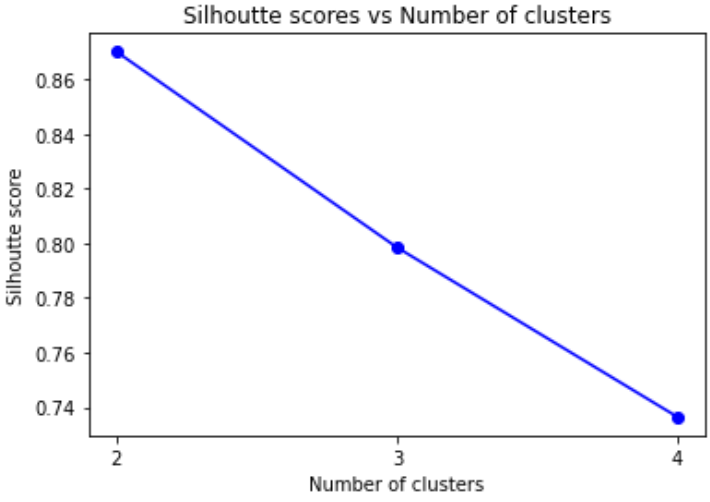
In [17]:

```
from sklearn.cluster import KMeans

kmeans_models = [KMeans(n_clusters=k, random_state=23).fit(X_red) for k in range (1, 10)
]
innertia = [model.inertia_ for model in kmeans_models]

plt.plot(range(1, 10), innertia)
plt.title('Elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



In [18]:

```
from sklearn.metrics import silhouette_score

silhoutte_scores = [silhouette_score(X_red, model.labels_) for model in kmeans_models[1:
4]]
plt.plot(range(2,5), silhoutte_scores, "bo-")
plt.xticks([2, 3, 4])
plt.title('Silhoutte scores vs Number of clusters')
plt.xlabel('Number of clusters')
plt.ylabel('Silhoutte score')
plt.show()
```



In [19]:

```
from sklearn.metrics import silhouette_score

kmeans = KMeans(n_clusters=2, random_state=23)
kmeans.fit(X_red)

print('Silhoutte score of our model is ' + str(silhouette_score(X_red, kmeans.labels_)))
```

```
Silhoutte score of our model is 0.8700455999561806
```

In [20]:

```
df['cluster_id'] = kmeans.labels_
```

In [21]:

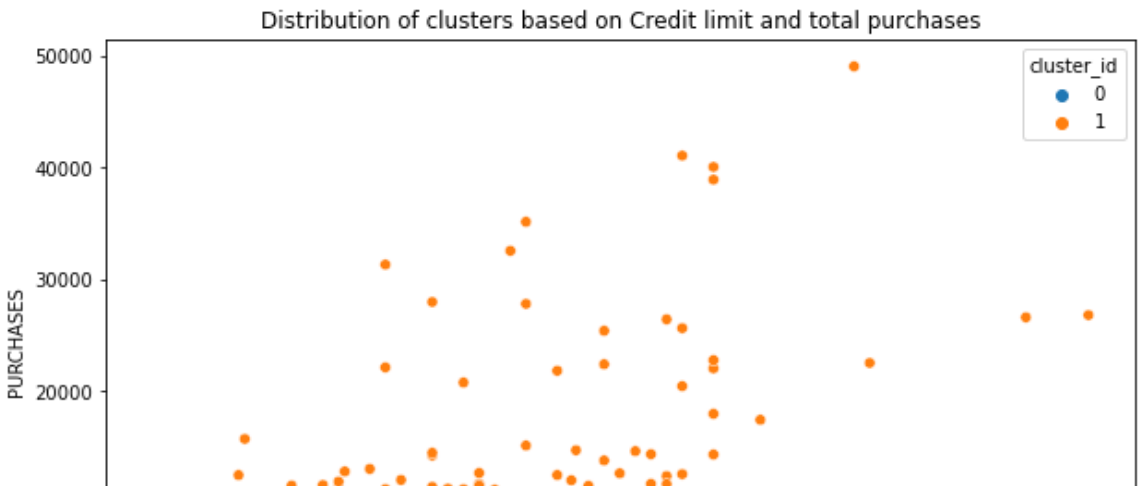
```
for col in cols:
    df[col] = np.exp(df[col])
```

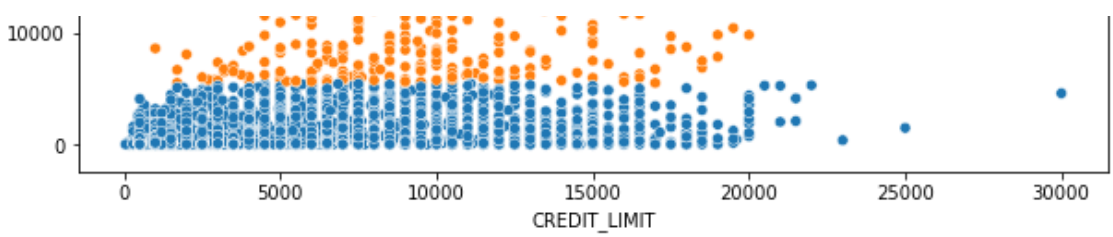
In [22]:

```
plt.figure(figsize=(10,6))
sns.scatterplot(data=df, x='ONEOFF_PURCHASES', y='PURCHASES', hue='cluster_id')
plt.title('Distribution of clusters based on One off purchases and total purchases')
plt.show()
```

In [23]:

```
plt.figure(figsize=(10,6))
sns.scatterplot(data=df, x='CREDIT_LIMIT', y='PURCHASES', hue='cluster_id')
plt.title('Distribution of clusters based on Credit limit and total purchases')
plt.show()
```




10000
0
0
5000
10000
15000
20000
25000
30000
CREDIT_LIMIT