Business Problem

The problem here is figuring out how to group the bank's customers into different categories so that the bank can run better and more focused marketing campaigns. The bank has a lot of data about how customers use their credit cards, how much they spend, and how they make payments. The idea is to use this data to find distinct groups of customers who behave in similar ways. By understanding these groups, the bank can learn what different types of customers need and prefer. This will help the marketing team create more personalized ads and offers that speak directly to each group, making customers more engaged, increasing sales, and using marketing budgets more wisely.

Importing Libraries

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

Loading the Dataset

In [ ]:
```
df = pd.read_csv('Marketing_data.csv')
df.head()
```

Out[ ]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADV
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.0
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.9
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.0
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.7
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.0

In [ ]:
```
df.shape
```

Out[ ]:
```
(8950, 18)
```

Descritpion of features

CUSTID: *Unique identifier for a credit card customer*
BALANCE: *Remaining balance in the customer's account available for transactions*
BALANCE_FREQUENCY: *Frequency with which the balance is updated, with a value between 0 and 1 (1 = frequently updated, 0 = rarely updated)*
PURCHASES: *Total amount spent on purchases using the account*
ONEOFFPURCHASES: *Highest single transaction amount made in a single purchase*
INSTALLMENTS_PURCHASES: *Total amount spent on purchases made in installments*
CASH_ADVANCE: *Amount of cash taken in advance by the customer*
PURCHASES_FREQUENCY: *Rate at which purchases are made, scored between 0 and 1 (1 = frequent purchases, 0 = infrequent purchases)*
ONEOFF_PURCHASES_FREQUENCY: *Rate at which one-time purchases occur (1 = frequent, 0 = infrequent)*
PURCHASES_INSTALLMENTS_FREQUENCY: *Frequency of purchases made in installments (1 = frequent, 0 = infrequent)*
CASH_ADVANCE_FREQUENCY: *How often cash advances are utilized*
CASH_ADVANCE_TRX: *Number of transactions where cash advance was used*
PURCHASES_TRX: *Number of transactions made for purchases*
CREDIT_LIMIT: *The credit limit assigned to the user*
PAYMENTS: *Total amount paid by the customer*
MINIMUM_PAYMENTS: *Minimum amount required to be paid by the customer*
PRC_FULL_PAYMENT: *Percentage of the total bill paid in full by the customer*
TENURE: *Duration of time the customer has held the credit card*

Dataset Exploration

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

Out[ ]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000
mean	1564.474828	0.877271	1003.204834	592.437371	411.067645	978.871112
std	2081.531879	0.236904	2136.634782	1659.887917	904.338115	2097.163877
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	128.281915	0.888889	39.635000	0.000000	0.000000	0.000000
50%	873.385231	1.000000	361.280000	38.000000	89.000000	0.000000
75%	2054.140036	1.000000	1110.130000	577.405000	468.637500	1113.821139
max	19043.138560	1.000000	49039.570000	40761.250000	22500.000000	47137.211760

Here are some general observations about the dataset:

For BALANCE, the average balance is about 1564.47. The balances of customers are quite different from each other, shown by a high standard deviation of 2081.53. The lowest balance is 0, and the highest is around 19,043.14, which means there's a big range of balances.

For BALANCE_FREQUENCY, the average is around 0.88. This tells us that most customers regularly maintain their balance. The median is 1, which means at least half of the customers always keep their balance.

Looking at PURCHASES, the average amount spent is 1003.20, but the numbers vary a lot, as the standard deviation is 2136.63. The highest amount spent is very high, at 49,039.57, which might mean there are some customers who spend a lot.

For ONEOFF_PURCHASES and INSTALLMENTS_PURCHASES, the average for one-off purchases is 592.44, and the highest is 40,761.25. Installment purchases have an average of 411.07, with the highest being 22,500. This means some customers are making big purchases either in one go or in installments.

Regarding CASH_ADVANCE, the average cash advance is 978.87, but again, there's a lot of variation, with a standard deviation of 2097.16. The highest cash advance is 47,132.11, which shows that some customers take a lot of cash advances.

For the FREQUENCY COLUMNS (like PURCHASES_FREQUENCY, ONEOFF_PURCHASES_FREQUENCY, etc.), the values are between 0 and 1, showing how often different activities happen (like purchases, one-off purchases, etc.). For example, the average purchases frequency is 0.49, meaning that, on average, purchases happen about half the time.

For CREDIT_LIMIT, the average is 4494.45, but it changes a lot across customers, with a standard deviation of 3638.82. The smallest credit limit is 50, and the largest is 30,000.

Looking at PAYMENTS, the average payment is 1733.14, but there's a big spread, with a standard deviation of 2895.06. The lowest payment is 0, and the highest recorded payment is 50,721.48.

For MINIMUM_PAYMENTS, there are some missing values (8637 out of 8950 are available), so that needs to be handled. The average minimum payment is 864.21, but there's a lot of variation with a standard deviation of 2372.45.

Lastly, for TENURE, most customers have been around for 12 years, with an average of 11.52 and a median of 12, which shows that many customers have been with the company for many years.

Here are some thoughts on customer segmentation and behavior:

The big differences and high standard deviations in things like purchases, payments, and credit limits suggest that there could be different types of customers. For example, there might be high spenders who make big purchases often, or those who frequently take out cash advances.

When it comes to risk assessment, the large variability in cash advances and payments could be a sign of customers who might be more risky. Customers who often take big cash advances or have irregular payments might be considered higher risk.

Looking at behavioral patterns, the frequency columns (like how often customers make purchases or take cash advances) can give us a good idea of how customers behave. Some customers might make regular purchases, while others might use cash advances more often. This can help in understanding their habits and possibly predicting future behavior.

Let's at two customers to see how they behave in extreme ways. One of them made a really big one-off purchase of $40,000, which means they like using their card for big stuff but don't really use cash advances. The other customer, though, took out 47,000 in cash advances, which shows they prefer getting cash rather than buying things with the card. By checking them out, we can figure out what kind of products or offers would work best for each of them.

In [ ]:
```
df[df['ONEOFF_PURCHASES'] == 40761.25]
```

Out[ ]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_AD
550	C10574	11547.52001	1.0	49039.57	40761.25	8278.32	558

In [ ]:
```
df['CASH_ADVANCE'].max()
```

Out[ ]:
```
np.float64(47137.21176)
```

In [ ]:
```
df[df['CASH_ADVANCE'] == 47137.21176]
```

Out[ ]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_A
2159	C12226	10905.05381	1.0	431.93	133.5	298.43	471

Customer with the biggest one-off purchase spent around $40,761.25 in a single transaction, likely on something big like a car. His/her balance is about 11,000, and often use their card for large purchases but hardly ever take cash advances. This suggests he/she might be more interested in offers related to big purchases or installment plans rather than cash advances.

On the other hand, the customer with the highest cash advance took out $47,137.21. His/her balance is around 10,905, and don't spend much on regular purchases but frequently rely on cash advances, with 123 transactions. This might mean he/she need quick access to cash, so he/she could benefit from products focused on liquidity or lower interest rates on cash advances.

Looking at these two customers helps to understand different spending habits. One prefers big one-off purchases, while the other depends on cash advances, allowing for better-targeted marketing and product offerings.

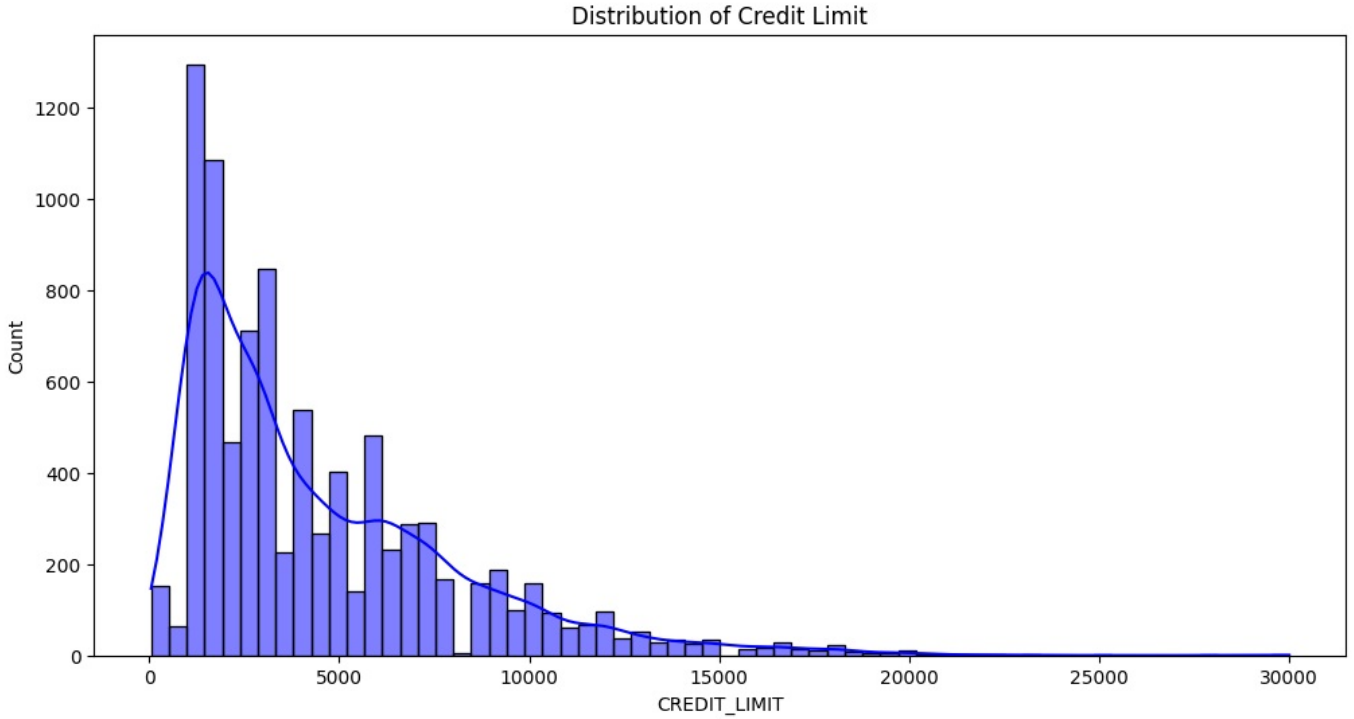
In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
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
```

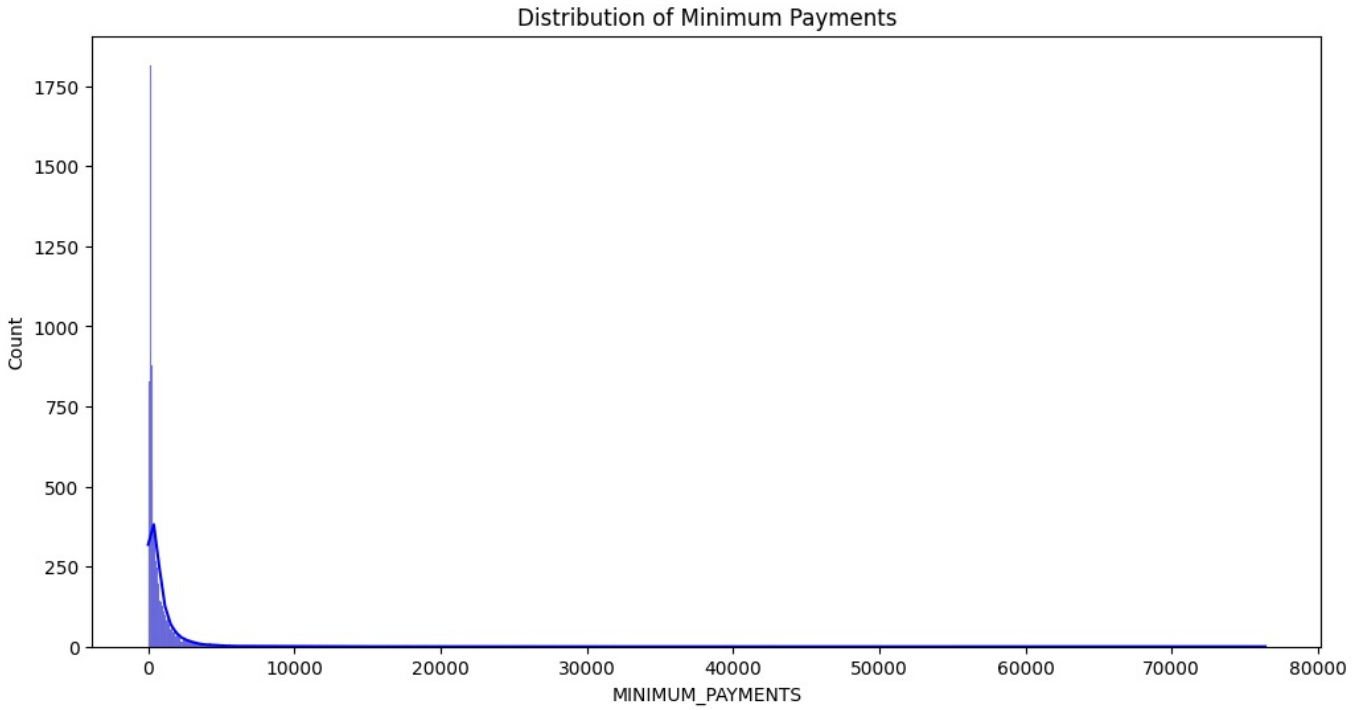
There are 1 missing data point in CREDIT_LIMIT and 313 missing data points in MINIMUM_PAYMENTS. Let's handle them, by reaplacing them with median value.

Based on the high standard deviations observed in columns like CREDIT_LIMIT and MINIMUM_PAYMENTS, it is likely that these columns have a skewed distribution or contain outliers. Thus, it would be safer to fill missing values using the median. But its more better to check distribution of these features to decide between median or mean with surety.

In [ ]:
```
plt.figure(figsize=(12,6))
plt.title("Distribution of Credit Limit")
sns.histplot(df['CREDIT_LIMIT'], color='blue', kde=True)
plt.show()
```



In [ ]:
```
plt.figure(figsize=(12,6))
plt.title("Distribution of Minimum Payments")
sns.histplot(df['MINIMUM_PAYMENTS'], color='blue', kde=True)
plt.show()
```



So its is confirmd that these two features are highly right skewed so it's better to use median

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

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

Out[ ]:
```
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
```

Let's drop CUST_ID column , becuase it of no use.

In [ ]:
```
df = df.drop('CUST_ID', axis=1)
```

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

Out[ ]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PUI
0	40.900749	0.818182	95.40	0.00	95.4	0.000000	
1	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	
2	2495.148862	1.000000	773.17	773.17	0.0	0.000000	
3	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	
4	817.714335	1.000000	16.00	16.00	0.0	0.000000	

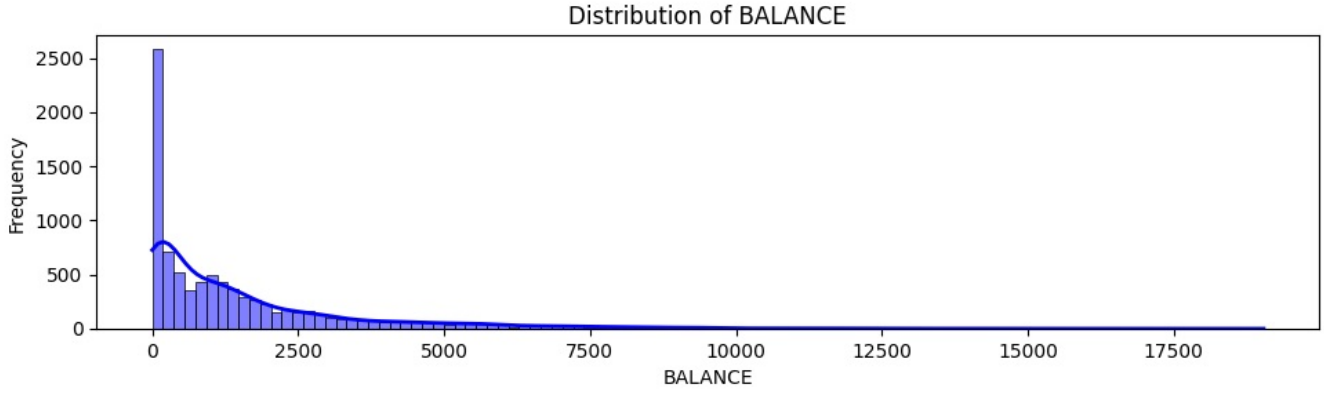
Let's check for duplica rows.

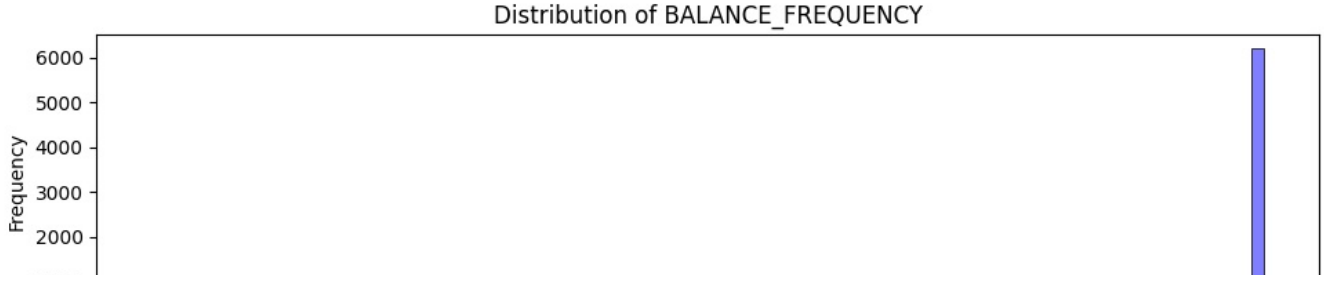
In [ ]:
```
df.duplicated().sum()
```

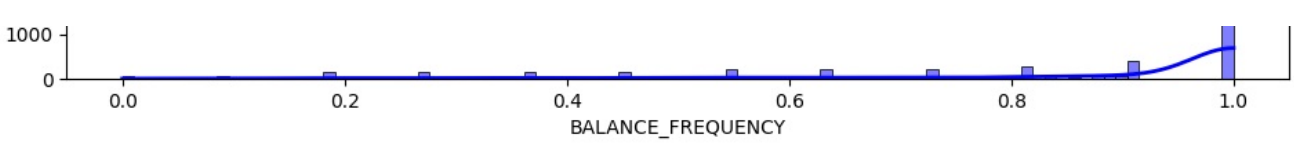
Out[ ]:
```
np.int64(0)
```

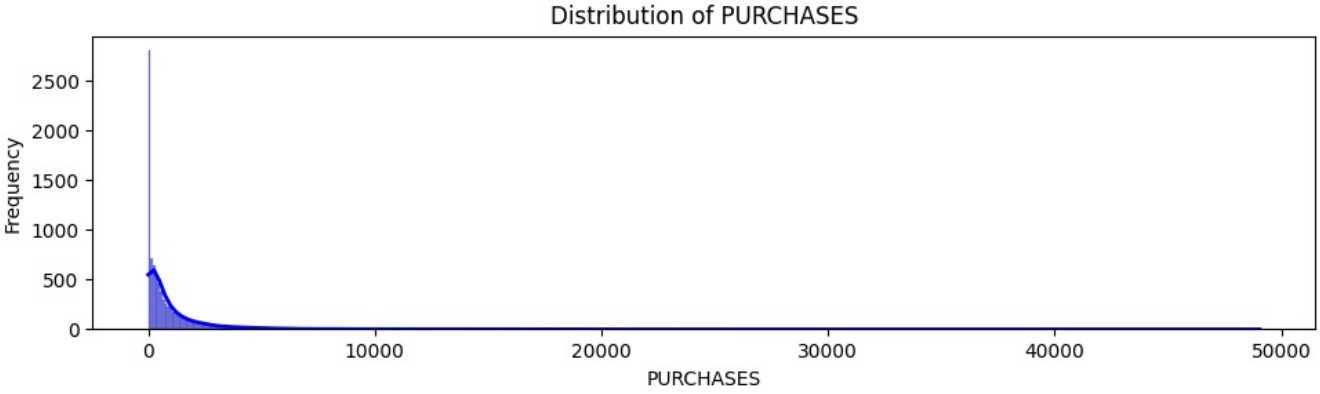
Now let's visualize distribution of all the columns.

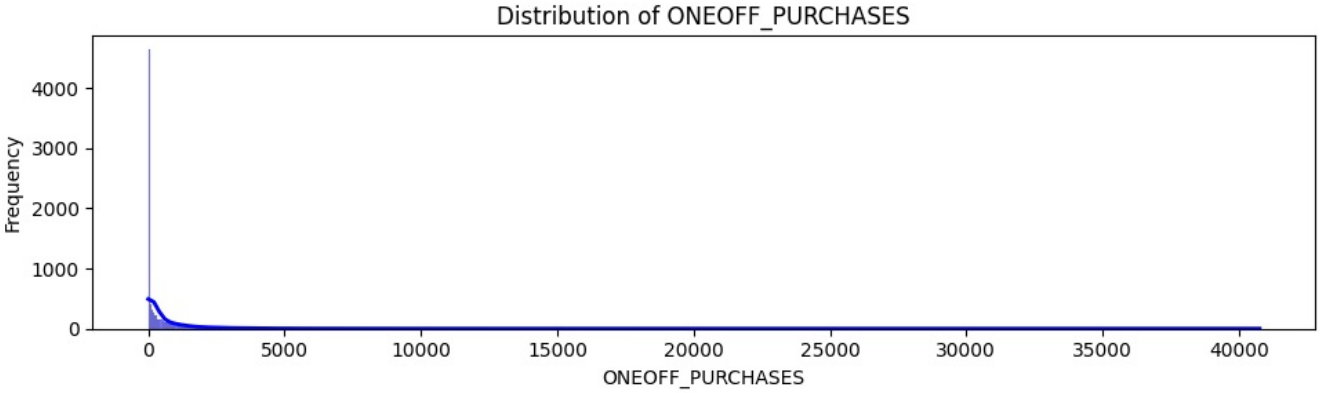
In [ ]:
```
plt.figure(figsize=(10,50))
for i in range(len(df.columns)):
  plt.subplot(17, 1, i+1)
  sns.histplot(df[df.columns[i]], kde=True, color='blue',line_kws={"lw": 2})
  plt.title(f"Distribution of {df.columns[i]}")
  plt.xlabel(df.columns[i])
  plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```

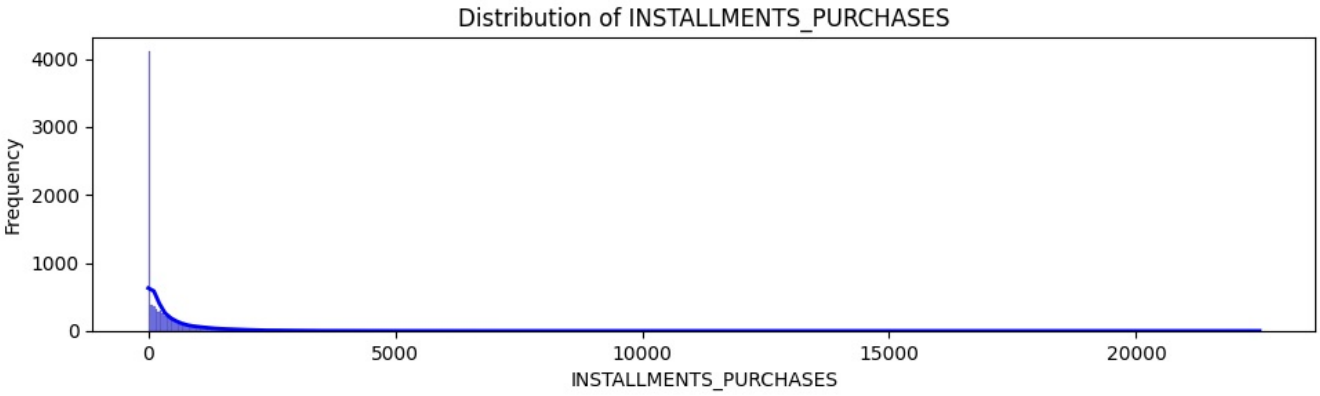


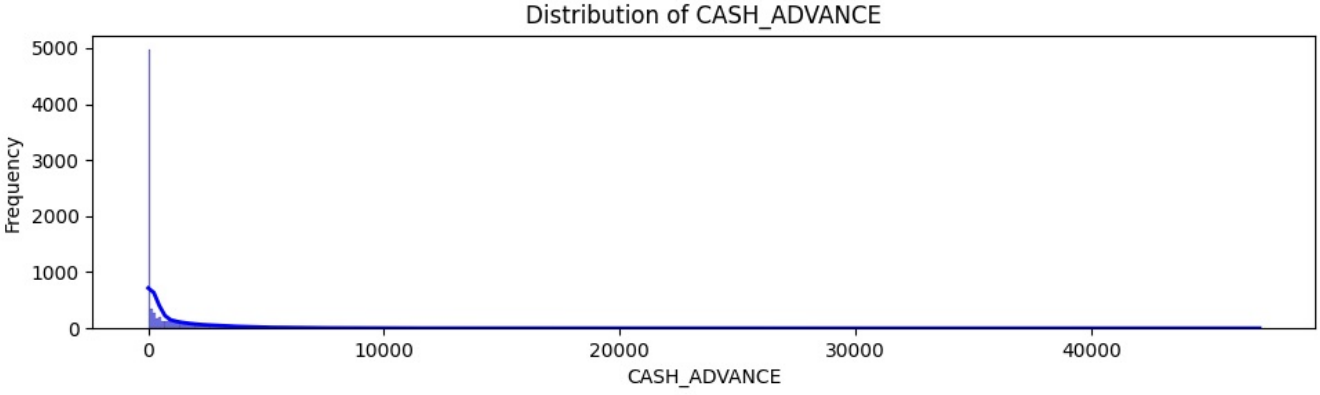


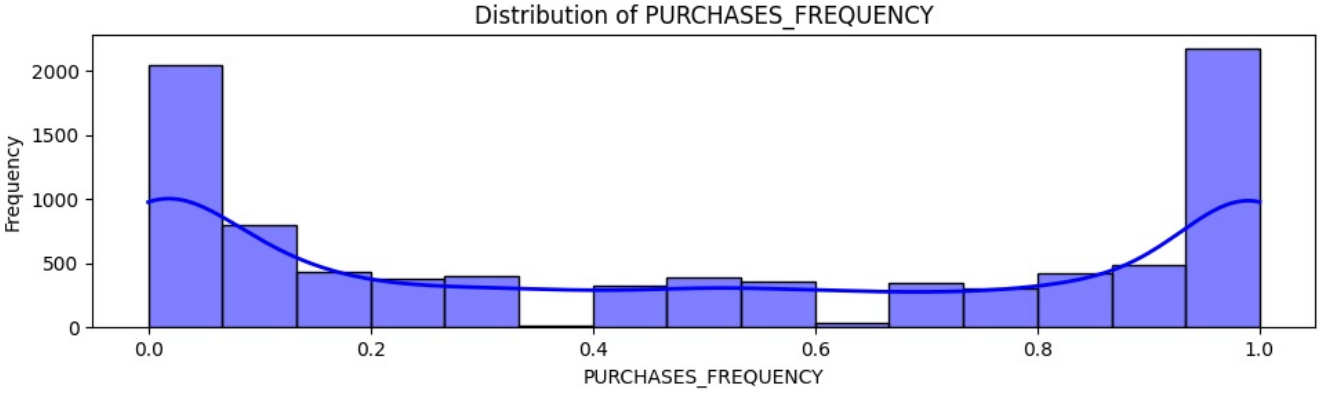

1000
0
0.0
0.2
0.4
0.6
0.8
1.0
BALANCE_FREQUENCY

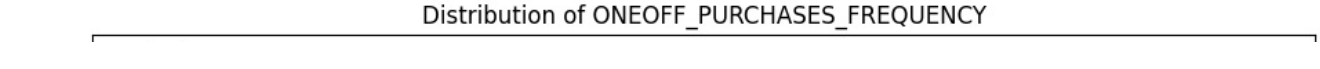

Distribution of PURCHASES
Frequency
2500
2000
1500
1000
500
0
0
10000
20000
30000
40000
50000
PURCHASES

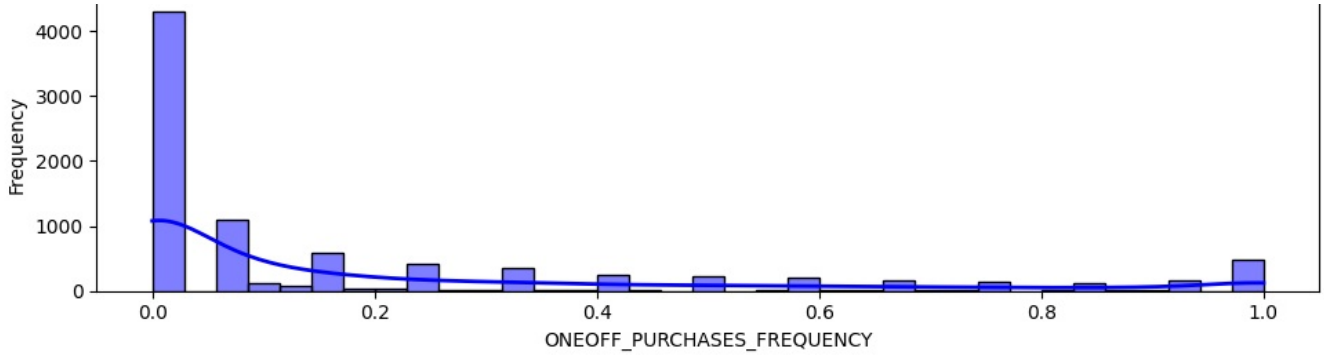

Distribution of ONEOFF_PURCHASES
Frequency
4000
3000
2000
1000
0
0
5000
10000
15000
20000
25000
30000
35000
40000
ONEOFF_PURCHASES

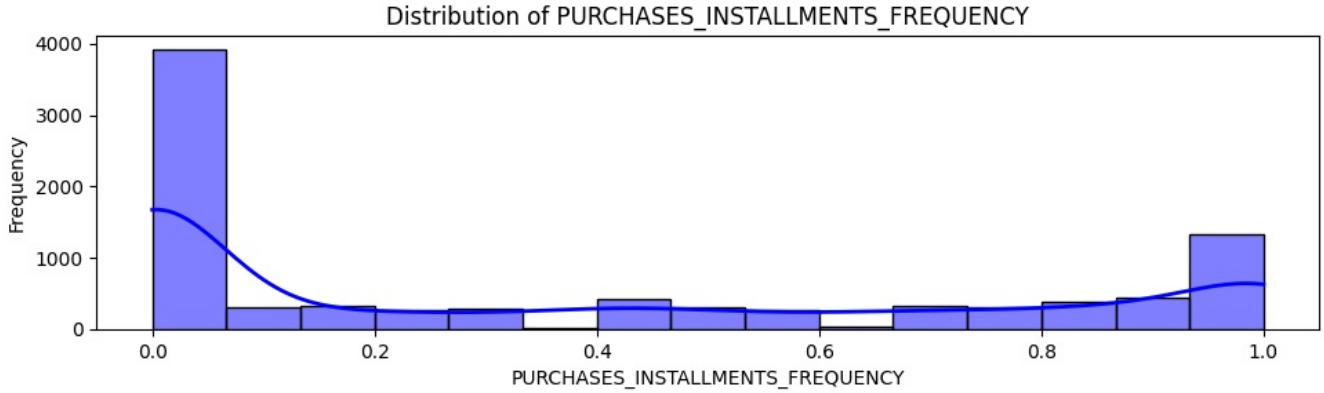

Distribution of INSTALLMENTS_PURCHASES
Frequency
4000
3000
2000
1000
0
0
5000
10000
15000
20000
INSTALLMENTS_PURCHASES

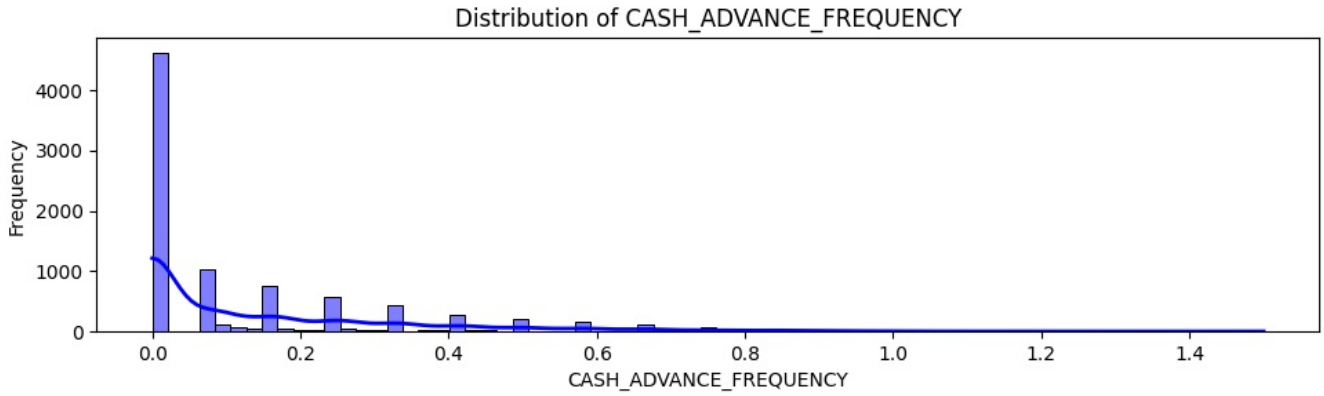

Distribution of CASH_ADVANCE
Frequency
5000
4000
3000
2000
1000
0
0
10000
20000
30000
40000
CASH_ADVANCE

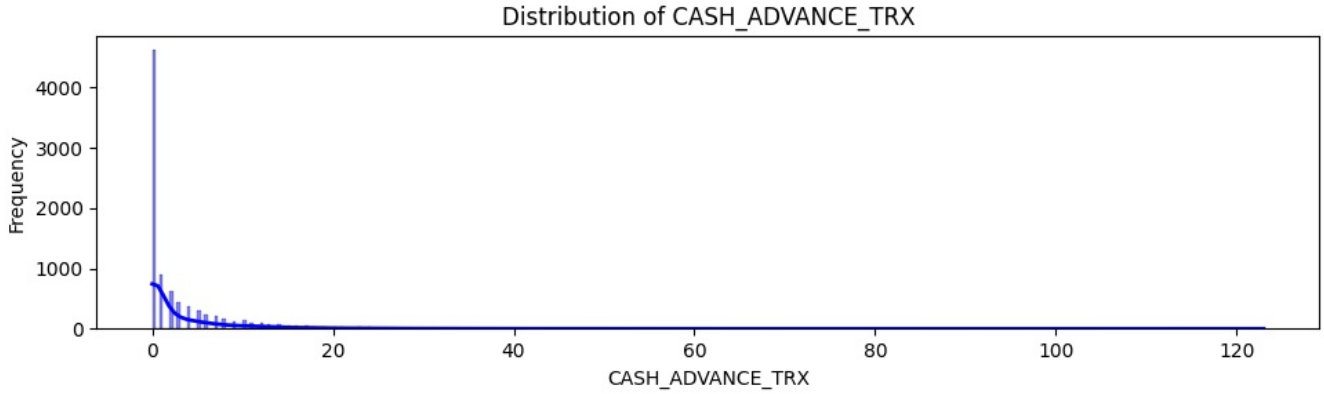

Distribution of PURCHASES_FREQUENCY
Frequency
2000
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
PURCHASES_FREQUENCY

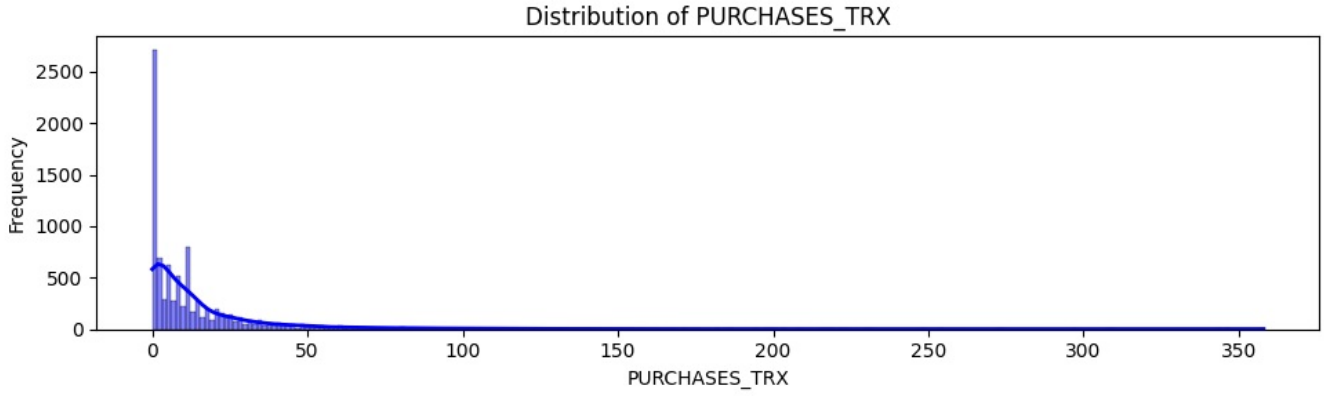

Distribution of ONEOFF_PURCHASES_FREQUENCY

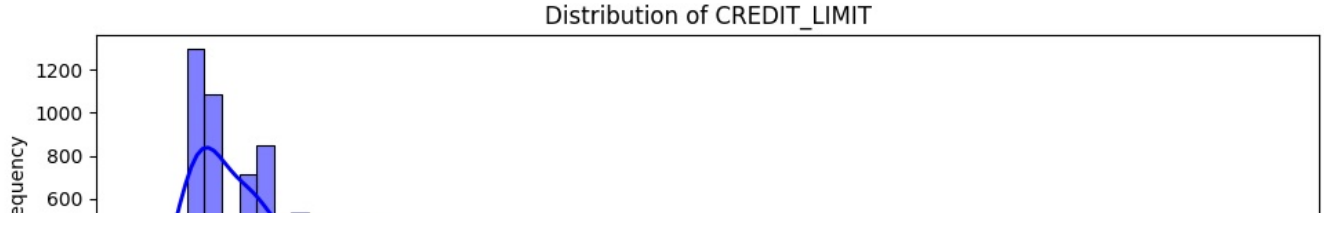

Frequency
ONEOFF_PURCHASES_FREQUENCY

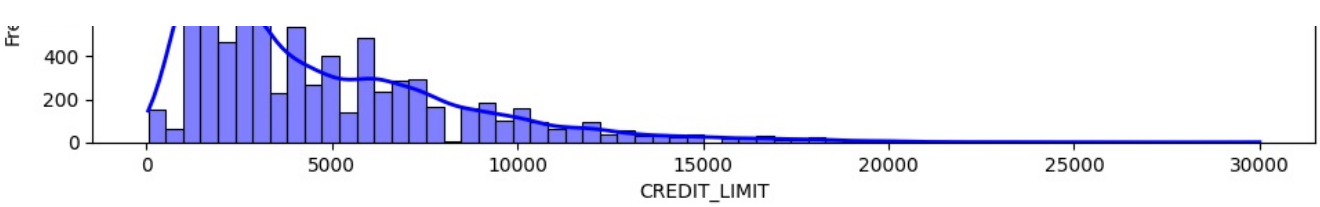

Distribution of PURCHASES_INSTALLMENTS_FREQUENCY
Frequency
PURCHASES_INSTALLMENTS_FREQUENCY

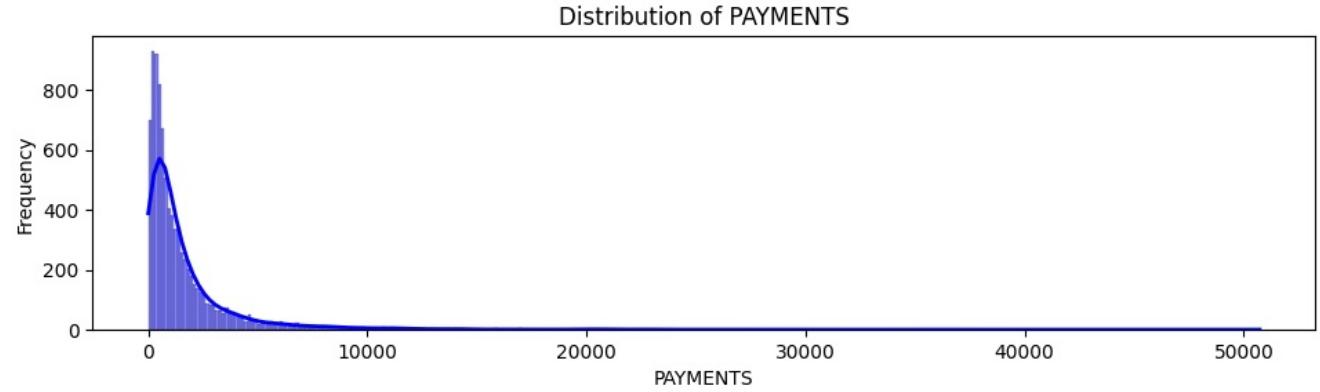

Distribution of CASH_ADVANCE_FREQUENCY
Frequency
CASH_ADVANCE_FREQUENCY

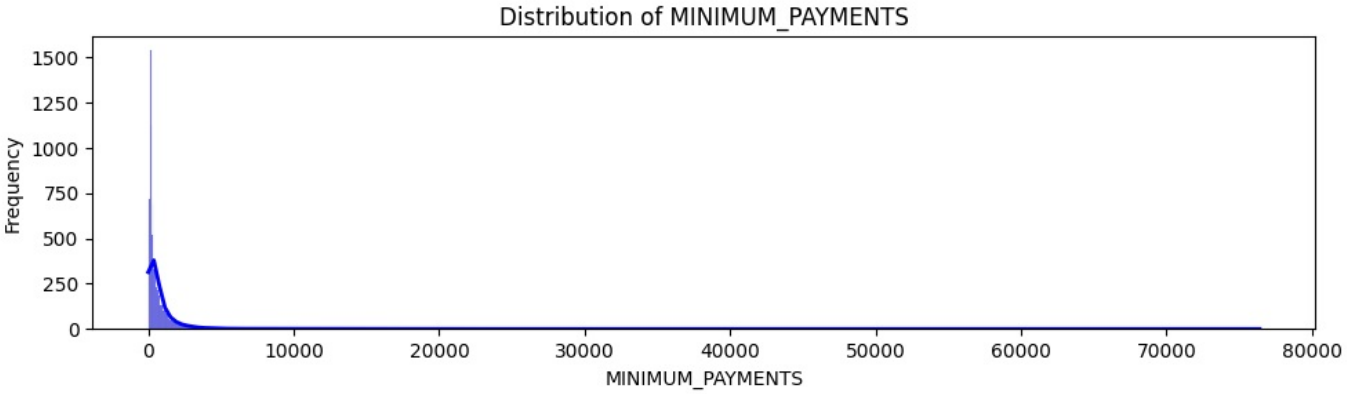

Distribution of CASH_ADVANCE_TRX
Frequency
CASH_ADVANCE_TRX

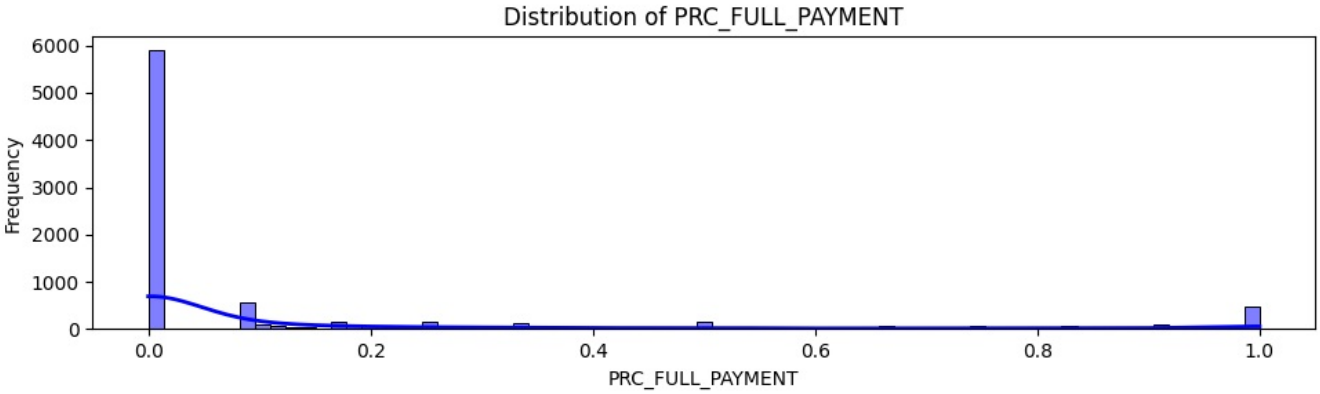

Distribution of PURCHASES_TRX
Frequency
PURCHASES_TRX

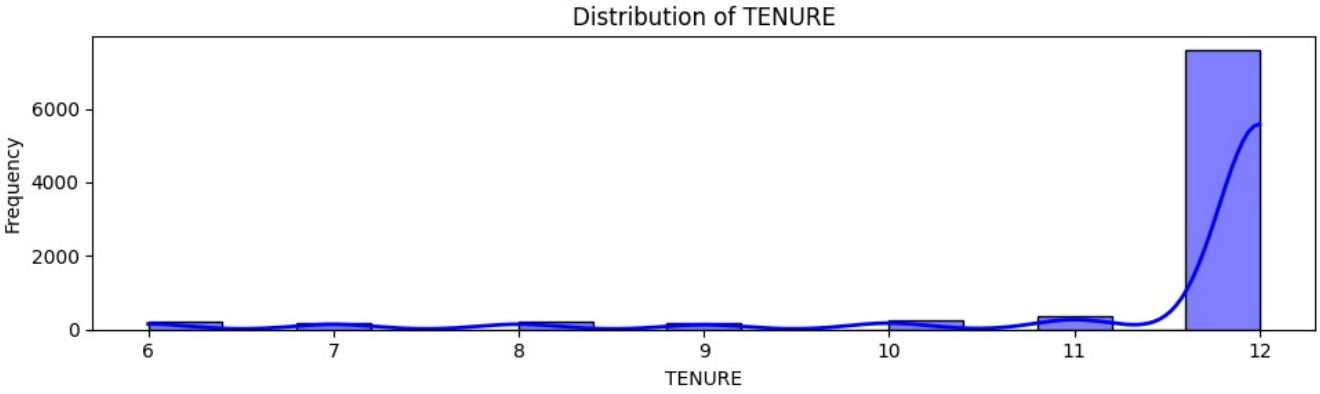

Distribution of CREDIT_LIMIT











For Balance, most customers have a balance of around $1,500. There are a few customers with much higher balances, like up to 20,000, but these are rare. This means most people keep their balances at a moderate level.

For Balance Frequency, most customers are at 1, which means they keep their balance steady and use their credit cards regularly, making frequent payments or updates.

When it comes to Purchases, most customers spend around $1,000 in total purchases. Some people spend a lot more, but those are just a few cases.

Looking at One-Off Purchases, most customers don't make large one-off purchases. They mostly spend small amounts, if at all, on these.

For Installment Purchases, it's the same story as one-off purchases. Most customers are not spending big amounts on installments.

In the Cash Advance category, most customers hardly use cash advances at all. The amounts are mostly close to zero, but a small group of customers takes out large cash advances.

For Purchases Frequency, there are two main groups of customers: those who use their credit cards for purchases all the time (frequency around 1) and those who don't use them as often. This shows that customers can be grouped based on how often they make purchases.

For One-Off Purchases Frequency, the chart shows that most people rarely do one-off purchases. This isn't a common behavior.

The Purchases Installments Frequency is pretty similar to the one-off purchases frequency—most people don't use installments often.

For Cash Advance Frequency, most customers have a frequency close to zero, so they rarely take out cash advances. This fits with what we saw with cash advance amounts—most people don't seem to need or want them.

The Cash Advance Transactions (TRX) chart shows most customers have only a few transactions involving cash advances, but a few have a lot.

For Purchases Transactions (TRX), most customers make a moderate number of purchase transactions. Some customers, though, make a lot more.

For Credit Limit, most customers have a limit between $1,000 and 10,000. There aren't too many people with very high credit limits, so those are less common.

Looking at Payments, most customers pay less than $2,000. However, a few people make really high payments, which stands out.

For Minimum Payments, most people have low minimum payments, but some have very high amounts, making them outliers.

When it comes to Percentage Full Payment (PRC_FULL_PAYMENT), most customers have a low percentage, between 0 and 0.2, meaning they don't pay their full balance. Only a small number pay their credit card in full.

Finally, for Tenure, most customers have been with the bank for 11 or 12 years. Not many have been there for less than six years, so most customers are pretty loyal and have been around for a long time.

The plots give us a lot of useful information about how customers use their credit cards, and this can help us figure out different groups of customers for targeted offers and products. For example, customers who use their credit cards often or have high balances might like deals on rewards programs or ways to manage their balance better. On the other hand, customers who take a lot of cash advances might be interested in products that offer lower fees or interest rates on cash advances. By understanding these patterns, we can create financial products that fit what different customers need and want.

Applying K-means Clustering

In [ ]:
```
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)
```

In [ ]:
```
df_scaled
```

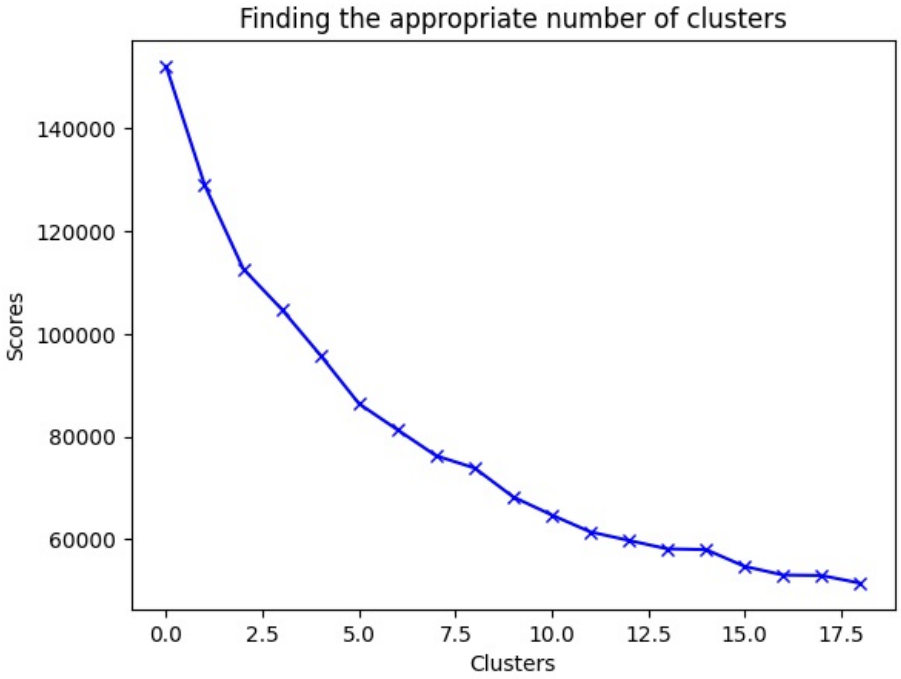
Out[ ]:
```
array([[-0.73198937, -0.24943448, -0.42489974, ..., -0.3024    ,
        -0.52555097,  0.36067954],
       [ 0.78696085,  0.13432467, -0.46955188, ...,  0.09749953,
         0.2342269 ,  0.36067954],
       [ 0.44713513,  0.51808382, -0.10766823, ..., -0.0932934 ,
        -0.52555097,  0.36067954],
       ...,
       [-0.7403981 , -0.18547673, -0.40196519, ..., -0.32687479,
         0.32919999, -4.12276757],
       [-0.74517423, -0.18547673, -0.46955188, ..., -0.33830497,
         0.32919999, -4.12276757],
       [-0.57257511, -0.88903307,  0.04214581, ..., -0.3243581 ,
        -0.52555097, -4.12276757]])
```

Let's find optimal number of clusters using Elbow Method

In [ ]:
```
score_1 = []
range_values = range(1,20)
for i in range_values:
  kmeans = KMeans(n_clusters=i)
  kmeans.fit(df_scaled)
  score_1.append(kmeans.inertia_)
```

In [ ]:
```
plt.plot(score_1, 'bx-')
plt.title('Finding the appropriate number of clusters')
plt.xlabel('Clusters')
plt.ylabel('Scores')
```

```
plt.show()
```



The Elbow Point on the graph is where we see the biggest drop in scores as we add more clusters, which happens from 1 to about 7 clusters. This means that adding more clusters up to 7 really helps to make the groups better by making them tighter and reducing the distance within each group. But after 7 clusters, the scores don't drop as much, and the line starts to flatten out. This creates an "elbow" shape at around 7 clusters.

The "elbow" point is where adding more clusters doesn't really make things much better, so it's not worth it to keep adding more clusters beyond this point.

The best number of clusters seems to be around 7. This is where the "elbow" is on the graph, showing that 7 clusters give a good balance. It helps make the groups clear without overcomplicating things by having too many clusters that might not be very useful.

Let's continue applying K-means clustering

In [ ]:
```
kmeans = KMeans(7)
kmeans.fit(df_scaled)
labels = kmeans.labels_
```

In [ ]:
```
kmeans.cluster_centers_.shape
```

Out[ ]:
```
(7, 17)
```

In [ ]:
```
cluster_centers = pd.DataFrame(data = kmeans.cluster_centers_, columns = [df.columns])
cluster_centers
```

Out[ ]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURC
0	-0.694395	0.090754	0.086872	-0.041380	0.281001	-0.449883	
1	0.001748	0.372665	-0.365609	-0.242996	-0.417934	-0.055145	
2	1.923051	0.337717	11.212042	10.600367	7.033118	0.419625	
3	1.615263	0.377216	-0.214653	-0.145053	-0.241118	1.989602	
4	0.713886	0.464549	1.992395	1.674384	1.634849	-0.216206	
5	-0.695085	-2.173522	-0.326612	-0.242052	-0.327456	-0.296768	
6	-0.177452	0.385932	0.022859	-0.056522	0.157929	-0.316316	

In [ ]:
```
cluster_centers = scaler.inverse_transform(cluster_centers)
cluster_centers = pd.DataFrame(data = cluster_centers, columns = [df.columns])
cluster_centers
```

Out[ ]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PU
0	119.150057	0.898769	1188.807927	523.755035	665.173195	35.446401	
1	1568.113840	0.965552	222.075330	189.114192	33.134902	863.229758	
2	5567.142164	0.957273	24957.905000	18186.875667	6771.029333	1858.844605	
3	4926.508920	0.966630	544.595823	351.679258	193.027437	5151.159978	
4	3050.367871	0.987318	5259.987675	3371.572239	1889.441077	525.477269	
5	117.713546	0.362383	305.393725	190.680891	114.953158	356.534100	
6	1195.122637	0.968694	1052.044382	498.621786	553.880524	315.540819	

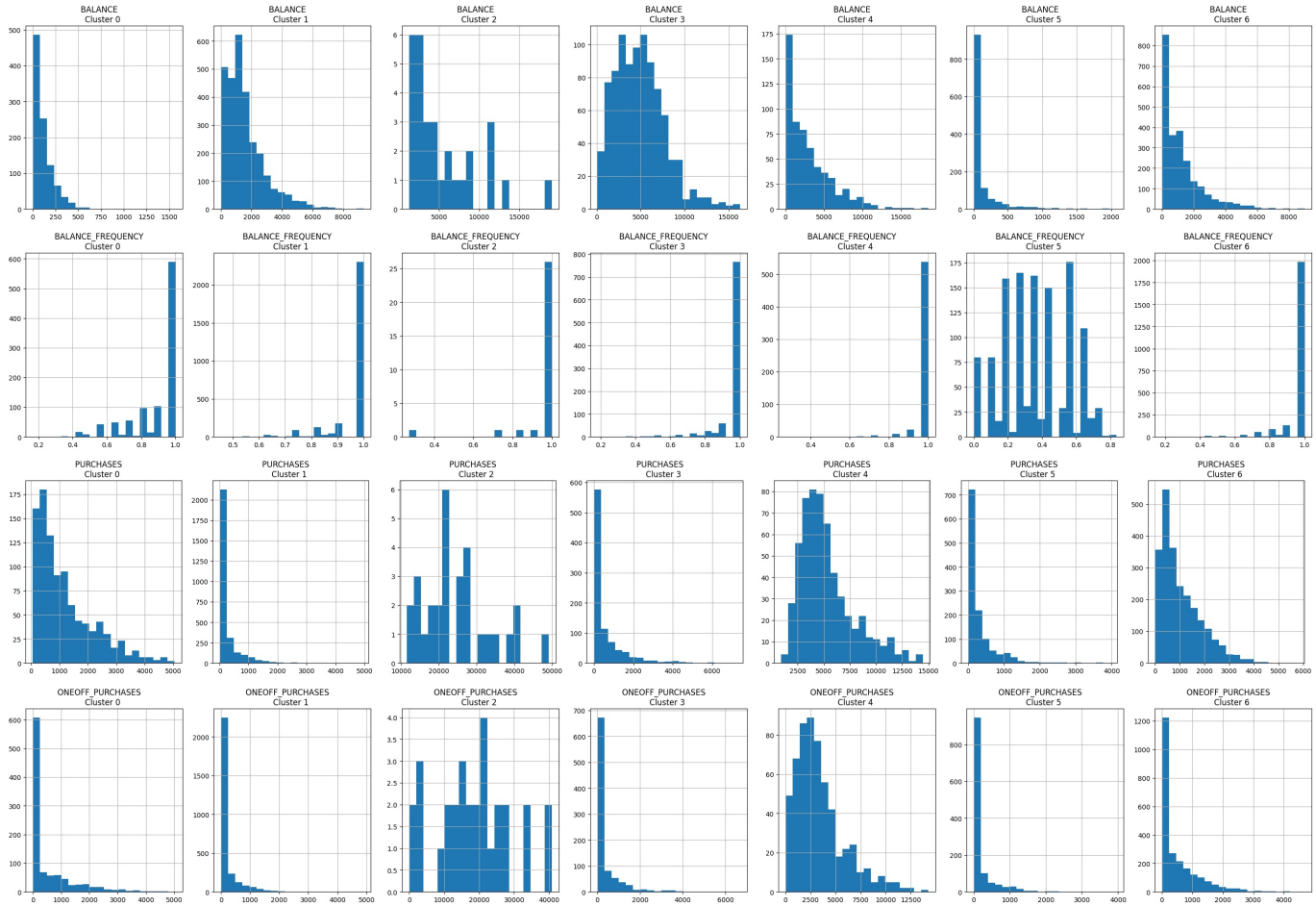
In [ ]:
```
df_cluster = pd.concat([df, pd.DataFrame({'cluster':labels})], axis = 1)
df_cluster.head()
```

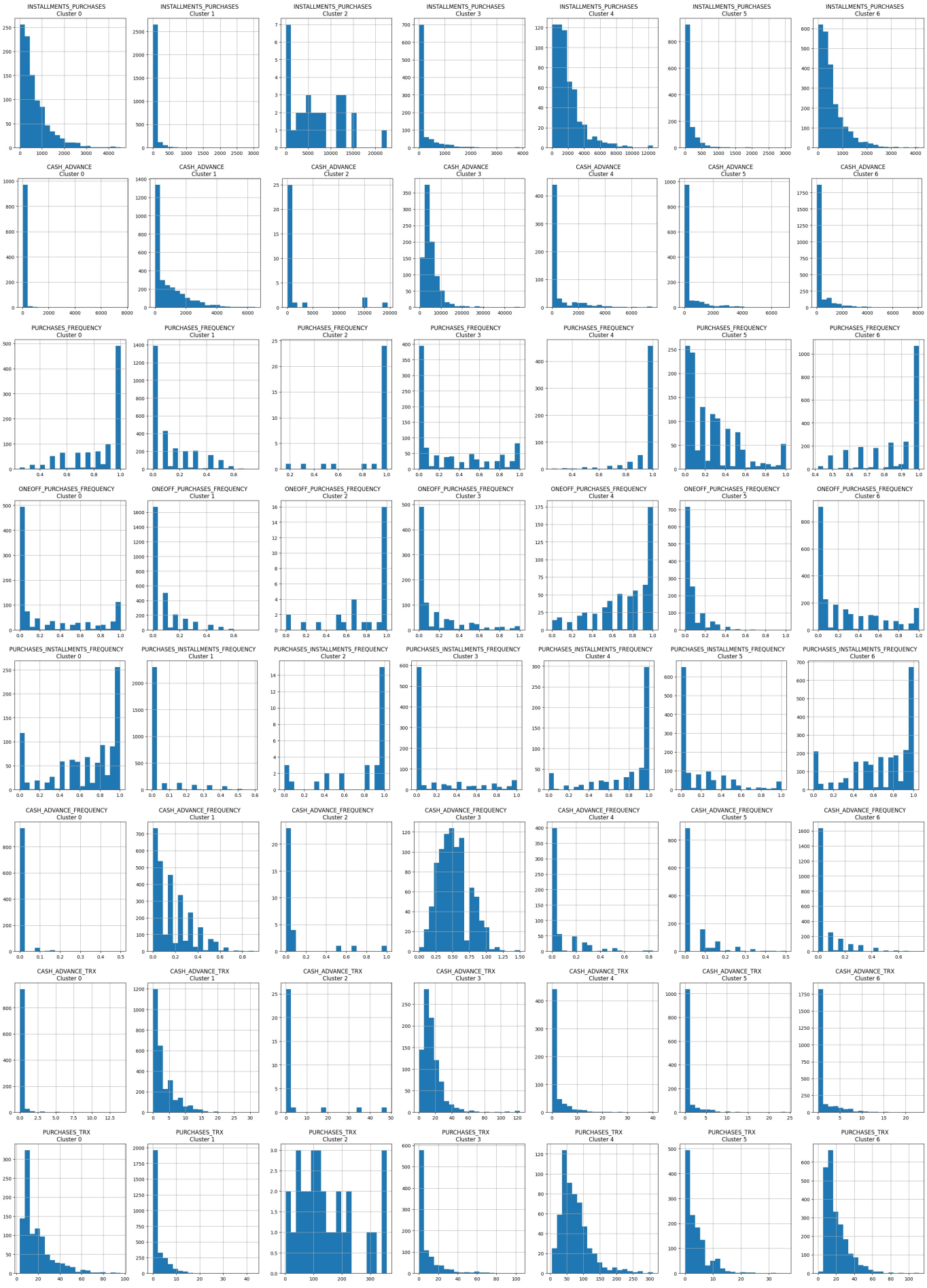
Out[ ]:

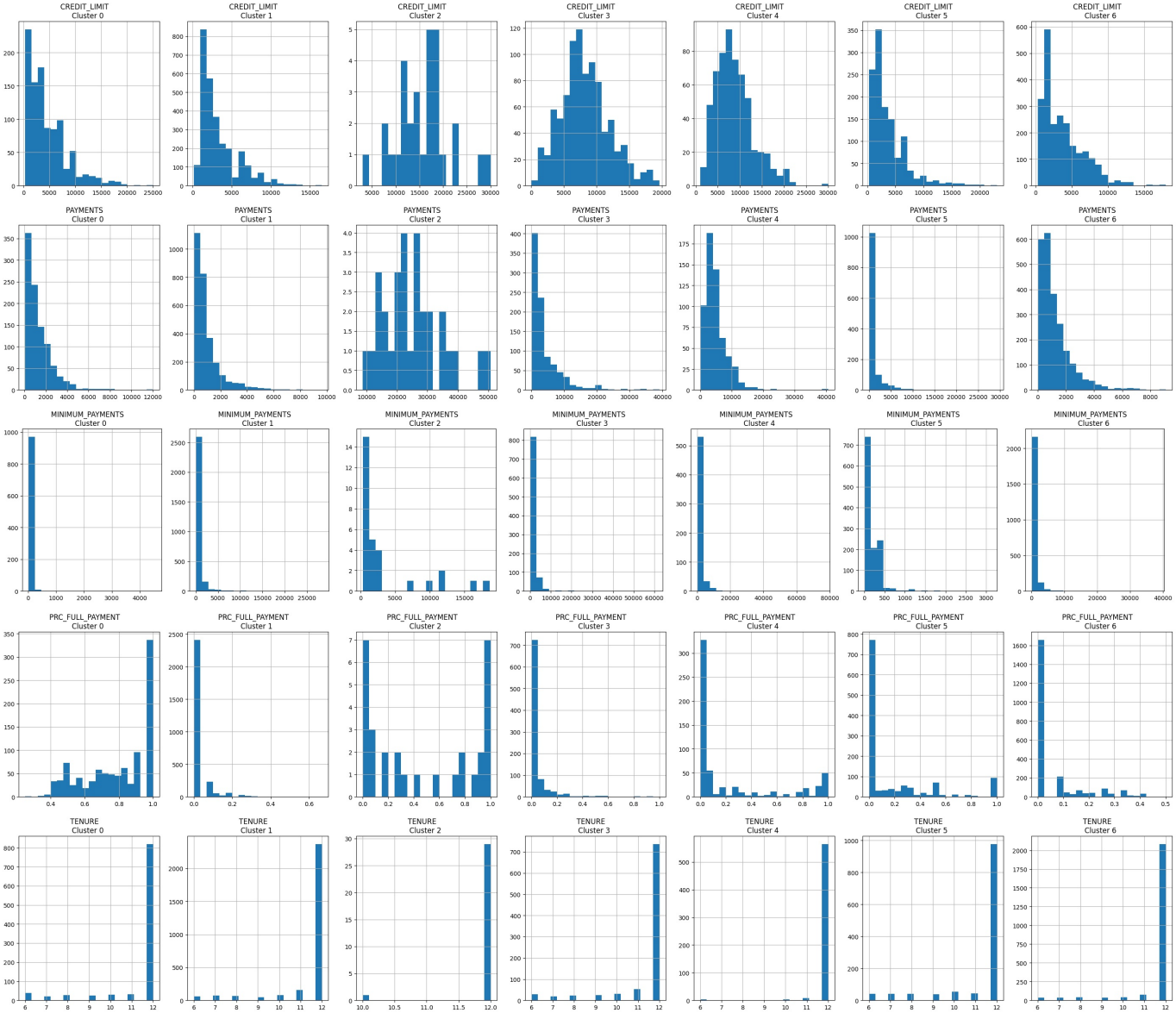
	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PU
0	40.900749	0.818182	95.40	0.00	95.4	0.000000	
1	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	
2	2495.148862	1.000000	773.17	773.17	0.0	0.000000	
3	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	
4	817.714335	1.000000	16.00	16.00	0.0	0.000000	

In [ ]:
```
for i in df.columns:
  plt.figure(figsize = (35, 5))
  for j in range(7):
    plt.subplot(1,7,j+1)
    cluster = df_cluster[df_cluster['cluster'] == j]
    cluster[i].hist(bins = 20)
    plt.title('{}    \nCluster {} '.format(i,j))

  plt.show()
```




INSTALLMENTS_PURCHASES Cluster 0
INSTALLMENTS_PURCHASES Cluster 1
INSTALLMENTS_PURCHASES Cluster 2
INSTALLMENTS_PURCHASES Cluster 3
INSTALLMENTS_PURCHASES Cluster 4
INSTALLMENTS_PURCHASES Cluster 5
INSTALLMENTS_PURCHASES Cluster 6
CASH_ADVANCE Cluster 0
CASH_ADVANCE Cluster 1
CASH_ADVANCE Cluster 2
CASH_ADVANCE Cluster 3
CASH_ADVANCE Cluster 4
CASH_ADVANCE Cluster 5
CASH_ADVANCE Cluster 6
PURCHASES_FREQUENCY Cluster 0
PURCHASES_FREQUENCY Cluster 1
PURCHASES_FREQUENCY Cluster 2
PURCHASES_FREQUENCY Cluster 3
PURCHASES_FREQUENCY Cluster 4
PURCHASES_FREQUENCY Cluster 5
PURCHASES_FREQUENCY Cluster 6
ONEOFF_PURCHASES_FREQUENCY Cluster 0
ONEOFF_PURCHASES_FREQUENCY Cluster 1
ONEOFF_PURCHASES_FREQUENCY Cluster 2
ONEOFF_PURCHASES_FREQUENCY Cluster 3
ONEOFF_PURCHASES_FREQUENCY Cluster 4
ONEOFF_PURCHASES_FREQUENCY Cluster 5
ONEOFF_PURCHASES_FREQUENCY Cluster 6
PURCHASES_INSTALLMENTS_FREQUENCY Cluster 0
PURCHASES_INSTALLMENTS_FREQUENCY Cluster 1
PURCHASES_INSTALLMENTS_FREQUENCY Cluster 2
PURCHASES_INSTALLMENTS_FREQUENCY Cluster 3
PURCHASES_INSTALLMENTS_FREQUENCY Cluster 4
PURCHASES_INSTALLMENTS_FREQUENCY Cluster 5
PURCHASES_INSTALLMENTS_FREQUENCY Cluster 6
CASH_ADVANCE_FREQUENCY Cluster 0
CASH_ADVANCE_FREQUENCY Cluster 1
CASH_ADVANCE_FREQUENCY Cluster 2
CASH_ADVANCE_FREQUENCY Cluster 3
CASH_ADVANCE_FREQUENCY Cluster 4
CASH_ADVANCE_FREQUENCY Cluster 5
CASH_ADVANCE_FREQUENCY Cluster 6
CASH_ADVANCE_TRX Cluster 0
CASH_ADVANCE_TRX Cluster 1
CASH_ADVANCE_TRX Cluster 2
CASH_ADVANCE_TRX Cluster 3
CASH_ADVANCE_TRX Cluster 4
CASH_ADVANCE_TRX Cluster 5
CASH_ADVANCE_TRX Cluster 6
PURCHASES_TRX Cluster 0
PURCHASES_TRX Cluster 1
PURCHASES_TRX Cluster 2
PURCHASES_TRX Cluster 3
PURCHASES_TRX Cluster 4
PURCHASES_TRX Cluster 5
PURCHASES_TRX Cluster 6



Cluster 0 can be described as regular users who have controlled spending habits. They keep a low balance, around 119, *buthaveahighbalancefrequencyof*0.89, *whichshowstheymanagetheircreditcardbalanceregularly*. *Theymakemoderatepurchases*, *roughly*1,189, which are split between one-off purchases of about 524*andinstallmentpurchasesaround*665. They use very little cash advance, just about 35, *andtheircashadvancefrequencyisalmostnegligibleat*0.006.*Theyhaveahighpurchasefrequencyof*0.85, *meaningtheyusetheircardsregularly*. *Withamoderate* 4,670, they make regular payments of about 1, 293*andkeeptheirminimumpaymentrequirementlowat*177. A high percentage of their full payment, 0.79, suggests they pay off their balances regularly. We could call these customers "Responsible Users" because they use their credit cards frequently but manage their balances well and often pay in full. They are ideal candidates for rewards programs and incentives for regular use.

Cluster 1 has low activity but is dependent on cash advances. They have a moderate balance of about 1, 568*andahighbalancefrequencyof*0.96, *whichshowstheykeeptheirbalancestable*. *However*, *theirpurchasesareverylow*, *onlyaround*222, and they don't use their card for purchases often, with a low purchase frequency of 0.13. They have high cash advance usage, approximately 863, *andtheytakecashadvancesrelativelyfrequently*, *withafrequencyof*0.17.*Theyrarelypayofftheirbalanceinfull*, *asindicatedbytheirverylowpercentageoffullpa* 3,299 and low overall payments of around $976. We could name this group "Cash Advance Reliant" since they depend more on cash advances rather than purchases. They might benefit from products offering lower fees for cash advances or financial counseling to reduce their reliance on them.

Cluster 2 consists of high spenders with maximum engagement. These customers keep a very high balance, around 5, 567, *andalsohaveahighbalancefrequencyof*0.96.*Theymakeextremelyhighpurchases*, *about*24,958, with significant amounts in one-off purchases of 18, 187*andinstallmentpurchasesofaround*6,771. Their cash advance usage is moderate, at about 1, 859, *withafrequencyof*0.08.*Theyusetheircardsoften*, *withapurchasefrequencyof*0.91, *andhavearelativelyhighpercentageoffullpaymentat*0.48.*Theircreditlin* 15,570, and they make large payments, averaging $25,178. This group can be named "Affluent High Spenders" because they are top-tier customers who spend heavily and use their credit cards for large transactions. They should be targeted with premium offers, high-value rewards, and exclusive services.

Cluster 3 represents high cash advance users with a low purchase frequency. These customers have a high balance of around 4, 927*andahighbalancefrequencyof*0.97.*Theymakelowpurchases*, *roughly*545, but have very high cash advance usage at about

5, 151*andahighcashadvancefrequencyof*0.52.*Theirpurchasefrequencyislowat*0.30, *andtheyhaveaverylowpercentageoffullpaymentat*0.04.*Theircreditlimitism*
8,086, but they have relatively high minimum payments, around $2,075. This cluster can be called "Cash Advance Heavy Users" because they rely heavily on cash advances and might be considered higher-risk customers. Products that help manage cash flow or reduce cash advance fees could be beneficial for them.

Cluster 4 is made up of moderate spenders who have balanced usage. They have a moderate balance of about 3, 050*andaveryhighbalancefrequencyof*0.98.*Theymakehighpurchases*, *around*5,260, which are balanced between one-off purchases of 3, 372*andinstallmentpurchasesofabout*1,889. Their cash advance usage is low, around 525, *withalowfrequencyof*0.07.*Theyhaveahighpurchasefrequencyof*0.95*andmakemoderatepayments*, *roughly*4,899. Their percentage of full payment is low at 0.23. We could name this group "Balanced Spenders" since they maintain a balanced approach to spending, with both one-off and installment purchases. They could be targeted with flexible rewards programs and promotional offers for regular purchases and installments.

Cluster 5 consists of inactive low spenders with minimal engagement. They have a very low balance, around 118, *andalowbalancefrequencyof*0.36.*Theymakelowpurchases*, *about*305, and their purchase frequency is very low, at 0.25. Their cash advance usage is moderate, at around 357, *withalowfrequencyof*0.04.*Theyhavealowcreditlimitofabout*3,612 and low payments of roughly $1,076. Their percentage of full payment is also low, at 0.17. This cluster can be named "Dormant Users" because these are inactive customers who rarely use their cards. They could be encouraged to use their cards more frequently with special offers, introductory rewards, or waived fees.

Cluster 6 includes regular spenders who have controlled usage. They maintain a low to moderate balance, around 1, 195, *andaveryhighbalancefrequencyof*0.97.*Theymakemoderatepurchasesofabout*1,052, with a mix of one-off purchases of 499*andinstallmentpurchasesofaround*554. They have low cash advance usage, roughly 316, *andalowfrequencyof*0.06.*Theyhaveahighpurchasefrequencyof*0.86*andlowcashadvancetransactionsofabout*1.16.*Theircreditlimitismoderate*, *around* 3,737, and their payments are also moderate, approximately $1,215. We could name this group "Regular and Controlled Spenders" because these customers make regular purchases and maintain good control over their usage. They could benefit from cashback offers, loyalty programs, and incentives for regular use.