Clustering for credit card data - customer segmentation

In [1]:

```
#importing importiant libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import cm as cm
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
from pandas.api.types import is_numeric_dtype
```

In [2]:

```
#loading data
df = pd.read_csv("CC GENERAL.csv")
```

In [3]:

```
df.head()
```

Out[3]:

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

In [4]:

```
#checking the range of data
df.describe()
```

Out[4]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	89
mean	1564.474828	0.877271	1003.204834	592.437371	411.067645	9
std	2081.531879	0.236904	2136.634782	1659.887917	904.338115	20
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	128.281915	0.888889	39.635000	0.000000	0.000000	
50%	873.385231	1.000000	361.280000	38.000000	89.000000	
75%	2054.140036	1.000000	1110.130000	577.405000	468.637500	11
max	19043.138560	1.000000	49039.570000	40761.250000	22500.000000	471

In [5]:

```
for col in df.columns:
    print(f'{col} : {df[col].nunique()} : {df[col].dtype}')
```

```
CUST_ID : 8950 : object
BALANCE : 8871 : float64
```

```
BALANCE_FREQUENCY : 43 : float64
PURCHASES : 6203 : float64
ONEOFF_PURCHASES : 4014 : float64
INSTALLMENTS_PURCHASES : 4452 : float64
CASH_ADVANCE : 4323 : float64
PURCHASES_FREQUENCY : 47 : float64
ONEOFF_PURCHASES_FREQUENCY : 47 : float64
PURCHASES_INSTALLMENTS_FREQUENCY : 47 : float64
CASH_ADVANCE_FREQUENCY : 54 : float64
CASH_ADVANCE_TRX : 65 : int64
PURCHASES_TRX : 173 : int64
CREDIT_LIMIT : 205 : float64
PAYMENTS : 8711 : float64
MINIMUM_PAYMENTS : 8636 : float64
PRC_FULL_PAYMENT : 47 : float64
TENURE : 7 : int64
```

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
```

Key Points: - data is all numerial, There are no categorical columns. -minimum value for mostly all columns is 0.0

In [7]:

```
#checking the shape of data
df.shape
```

Out[7]:

```
(8950, 18)
```

In [8]:

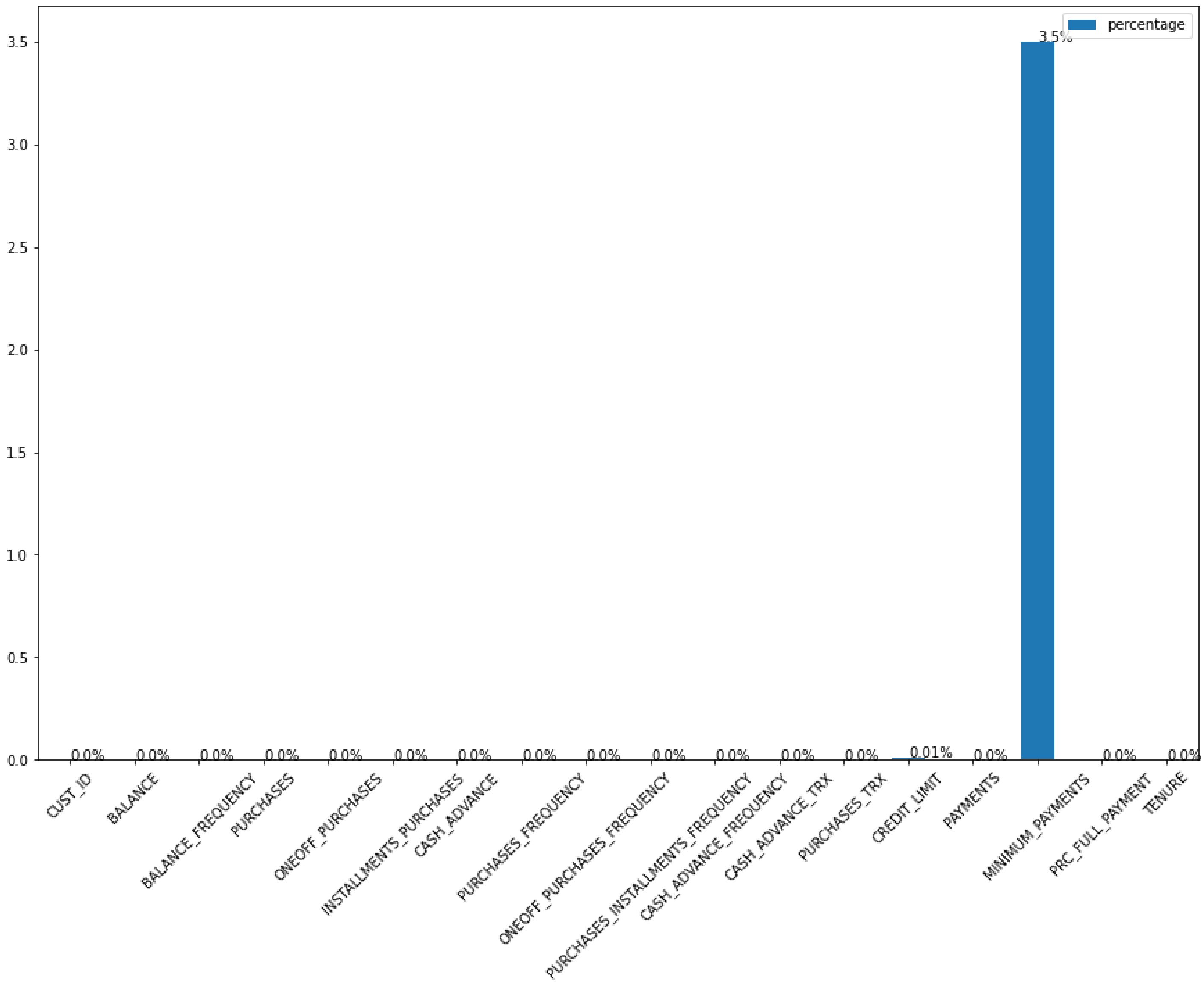
```
#checking the missing values
missing_count = df.isnull().sum()    # count of missing values
value_count = df.isnull().count()    # count of all values
missing_percentage = round(missing_count / value_count * 100,2 ) # percentage of missing v
missing_df = pd.DataFrame({'count':missing_count,'percentage': missing_percentage}) # crea
print(missing_df)
```

```
                                  count  percentage
CUST_ID                               0        0.00
BALANCE                               0        0.00
BALANCE_FREQUENCY                     0        0.00
PURCHASES                             0        0.00
```

```
ONEOFF_PURCHASES                     0        0.00
INSTALLMENTS_PURCHASES               0        0.00
CASH_ADVANCE                         0        0.00
PURCHASES_FREQUENCY                  0        0.00
ONEOFF_PURCHASES_FREQUENCY           0        0.00
PURCHASES_INSTALLMENTS_FREQUENCY     0        0.00
CASH_ADVANCE_FREQUENCY               0        0.00
CASH_ADVANCE_TRX                     0        0.00
PURCHASES_TRX                        0        0.00
CREDIT_LIMIT                         1        0.01
PAYMENTS                             0        0.00
MINIMUM_PAYMENTS                   313        3.50
PRC_FULL_PAYMENT                     0        0.00
TENURE                               0        0.00
```

In [9]:

```
# Plotting the bar chart for missing values


barchart = missing_df.plot.bar(y='percentage',rot=45,figsize=(15, 10))

for index,percentage in enumerate(missing_percentage):
    barchart.text(index,percentage,str(percentage) + "%")
```



In [10]:

```
# droping all rows with null values

df = df.dropna()
```

In [11]:

```
df.isnull().sum()
```

Out[11]:

```
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 [12]:

```
df.shape
```

Out[12]:

```
(8636, 18)
```

EDA and Feature engineering

In [13]:

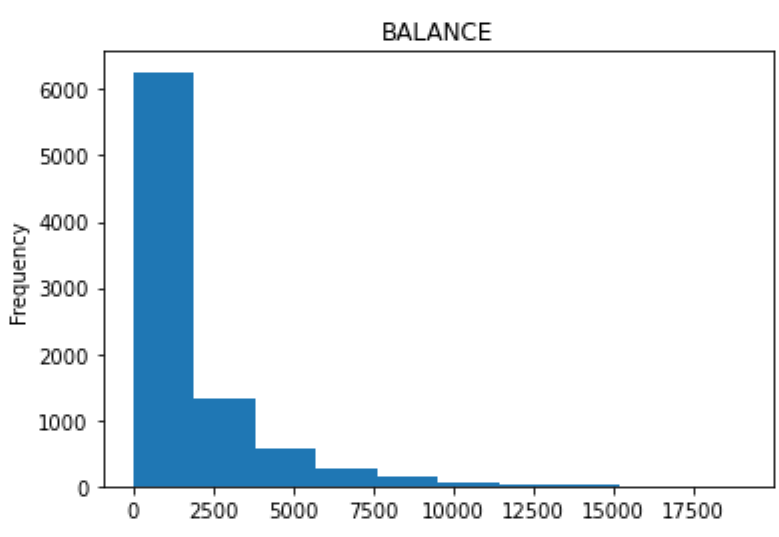
```
#dropping the columns which is not needed foe EDA
df = df.drop(['CUST_ID'],axis=1)
```

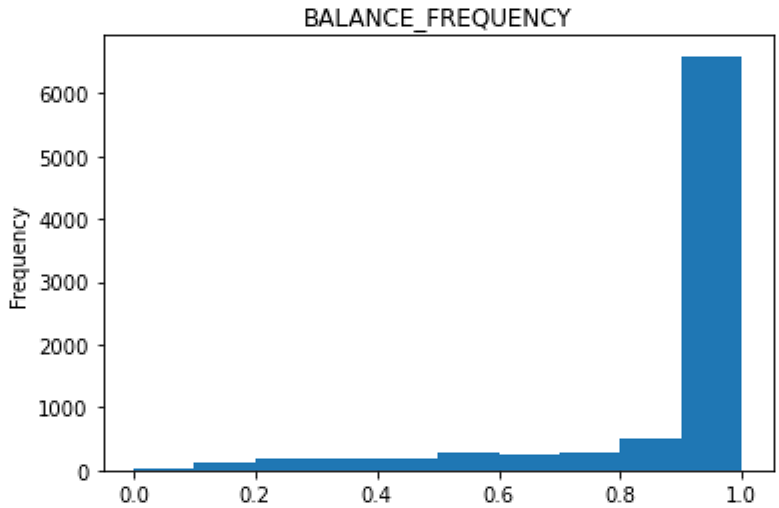
In [14]:

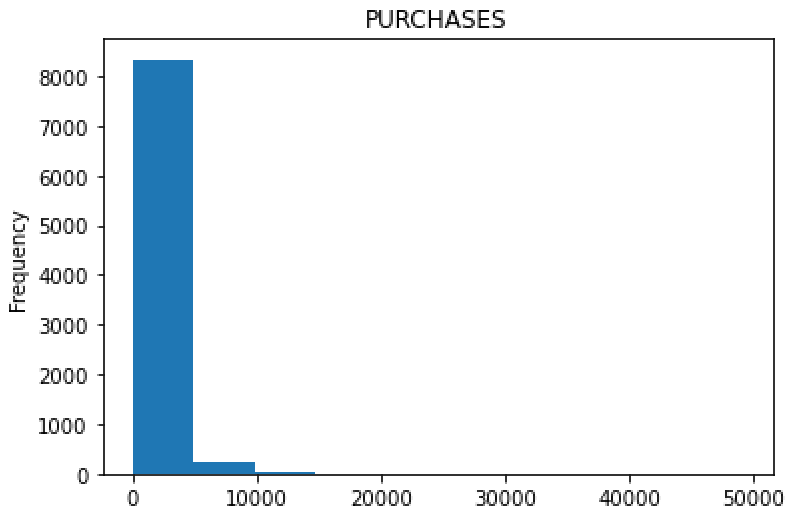
```
#Univariate analysis#
```

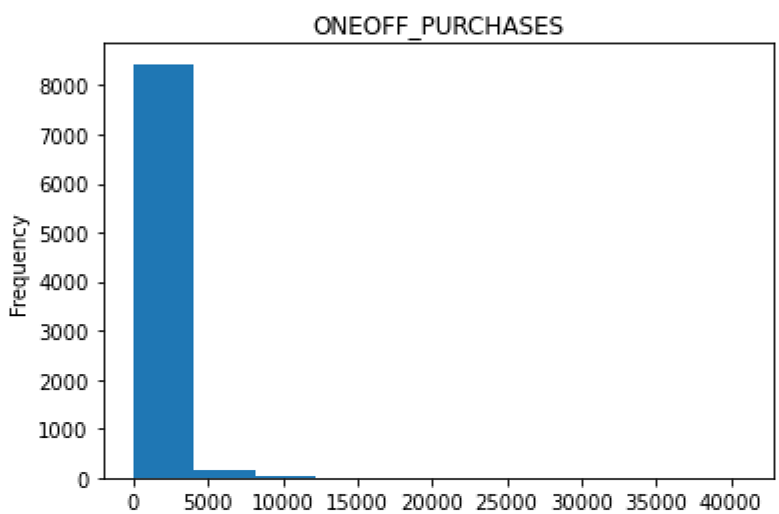
In [15]:

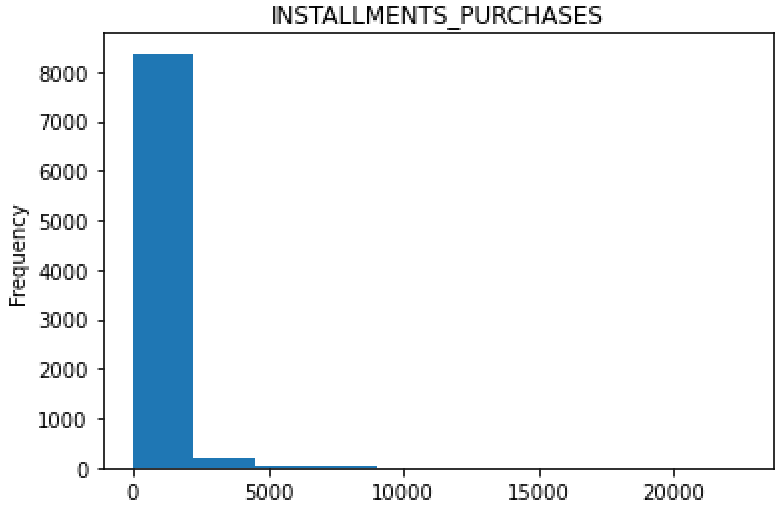
```
for column in df:
    plt.figure(column)
    plt.title(column)
    if is_numeric_dtype(df[column]):
        df[column].plot(kind='hist')
    else:
        print("No value")
```

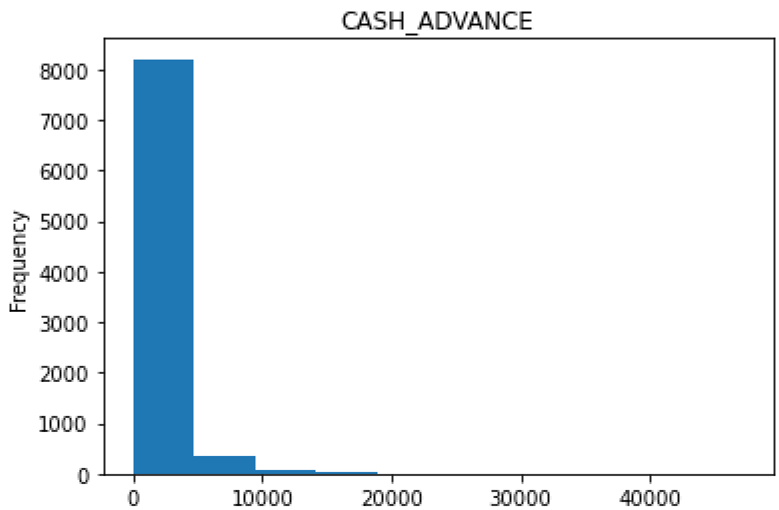


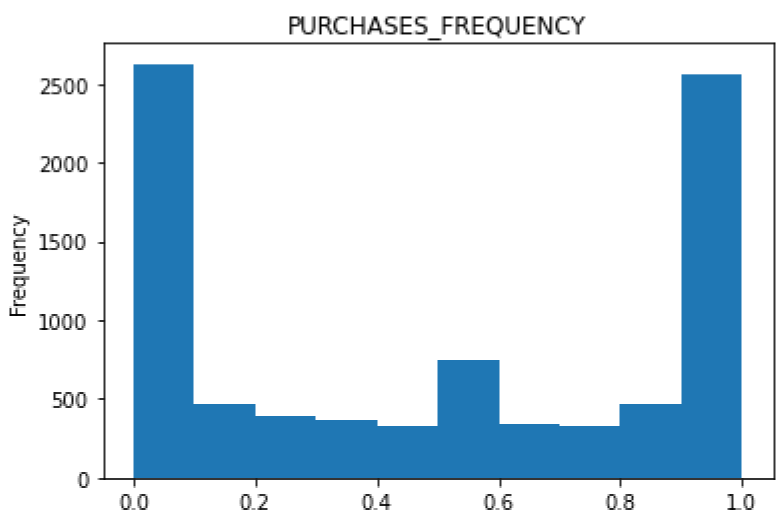

BALANCE_FREQUENCY
Frequency
0
1000
2000
3000
4000
5000
6000
0.0
0.2
0.4
0.6
0.8
1.0

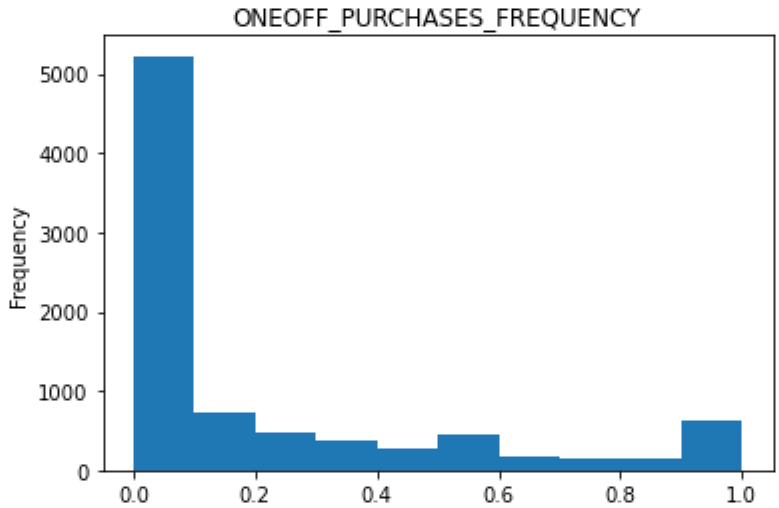

PURCHASES
Frequency
0
1000
2000
3000
4000
5000
6000
7000
8000
0
10000
20000
30000
40000
50000

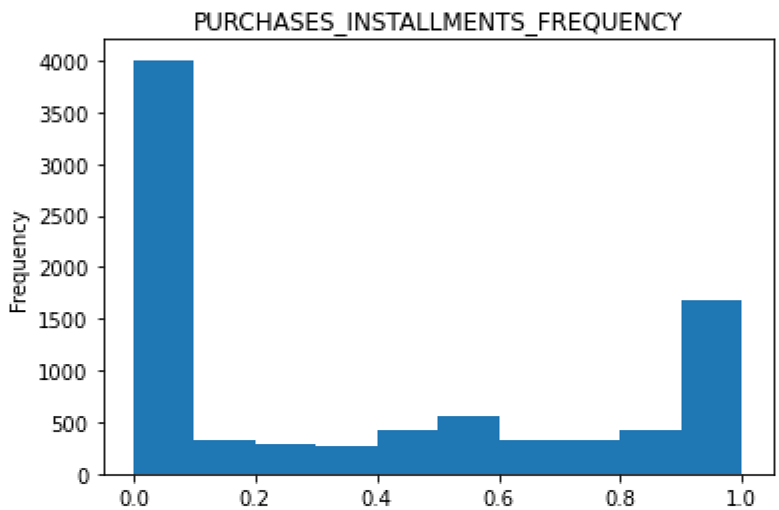

ONEOFF_PURCHASES
Frequency
0
1000
2000
3000
4000
5000
6000
7000
8000
0
5000
10000
15000
20000
25000
30000
35000
40000

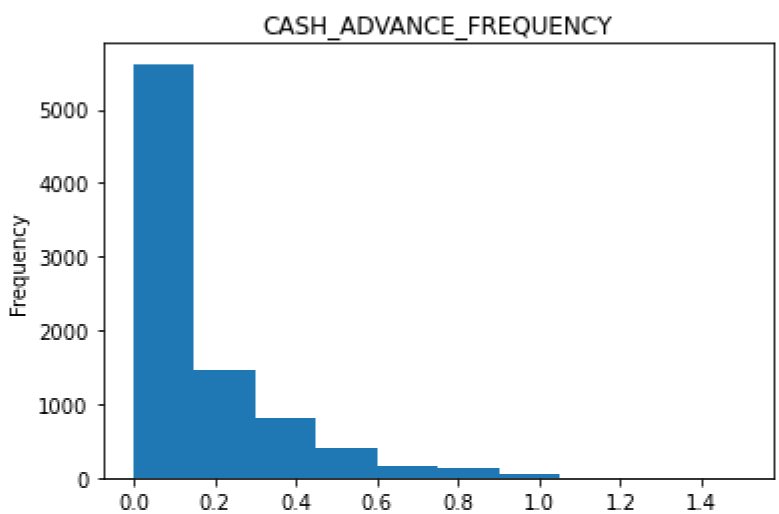

INSTALLMENTS_PURCHASES
Frequency
0
1000
2000
3000
4000
5000
6000
7000
8000
0
5000
10000
15000
20000

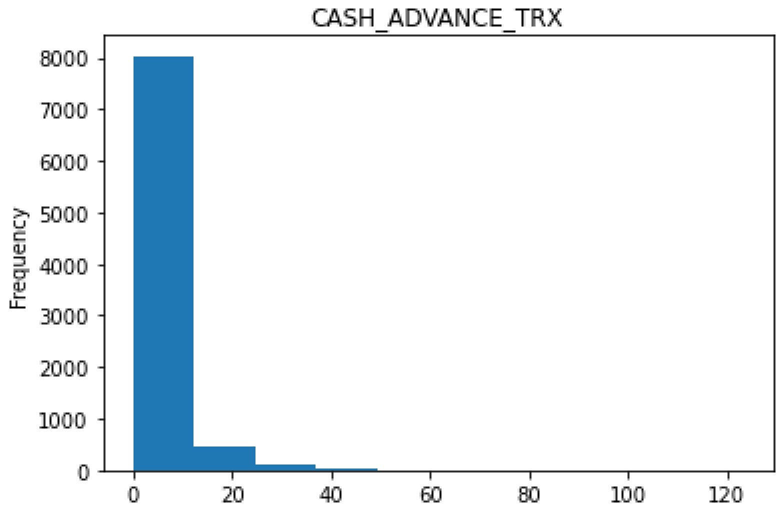

CASH_ADVANCE
Frequency
0
1000
2000
3000
4000
5000
6000
7000
8000
0
10000
20000
30000
40000

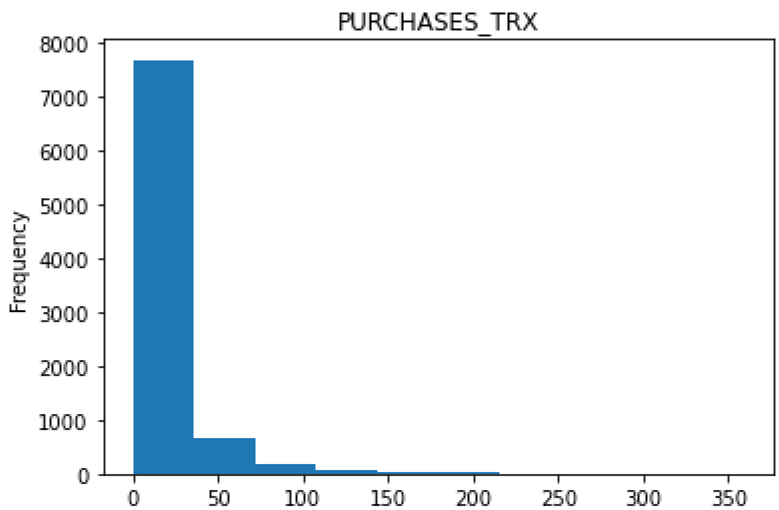

PURCHASES_FREQUENCY
Frequency
0
500
1000
1500
2000
2500
0.0
0.2
0.4
0.6
0.8
1.0

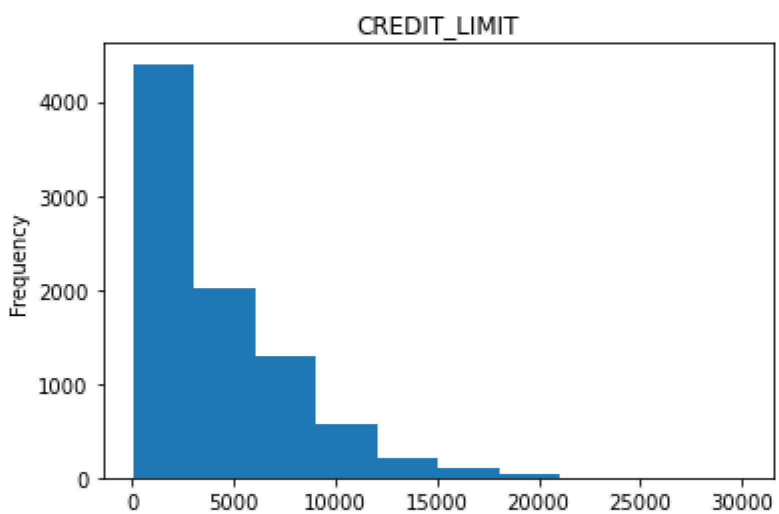

ONEOFF_PURCHASES_FREQUENCY
Frequency
0
1000
2000
3000
4000
5000
0.0
0.2
0.4
0.6
0.8
1.0

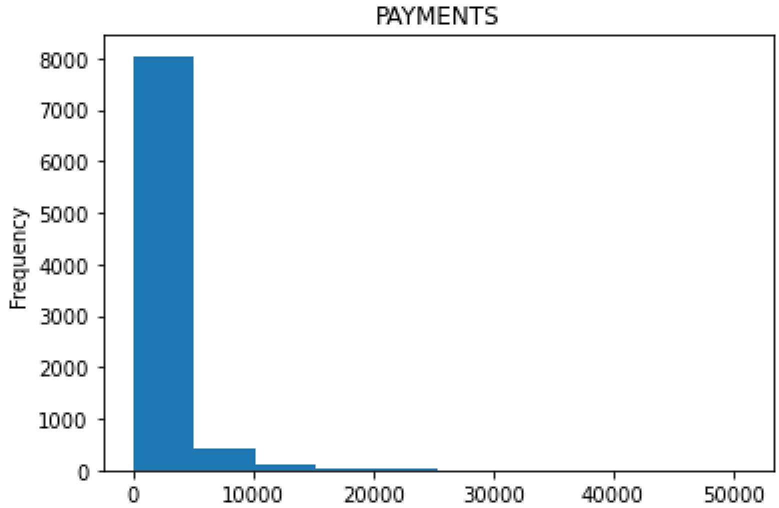

PURCHASES_INSTALLMENTS_FREQUENCY
Frequency
0
500
1000
1500
2000
2500
3000
3500
4000
0.0
0.2
0.4
0.6
0.8
1.0

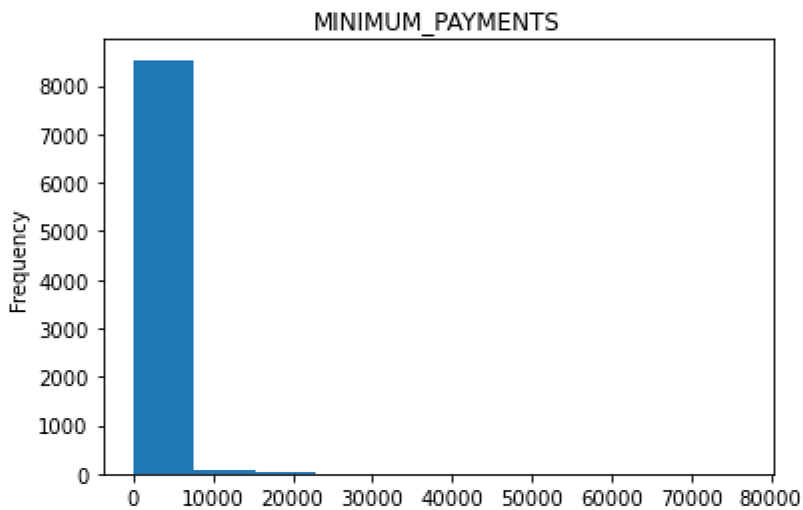

CASH_ADVANCE_FREQUENCY
Frequency
0
1000
2000
3000
4000
5000
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

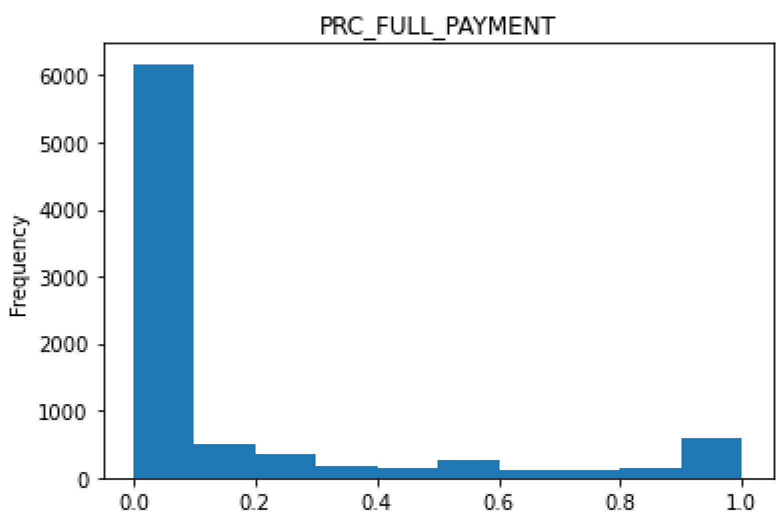

CASH_ADVANCE_TRX
Frequency
0
1000
2000
3000
4000
5000
6000
7000
8000
0
20
40
60
80
100
120

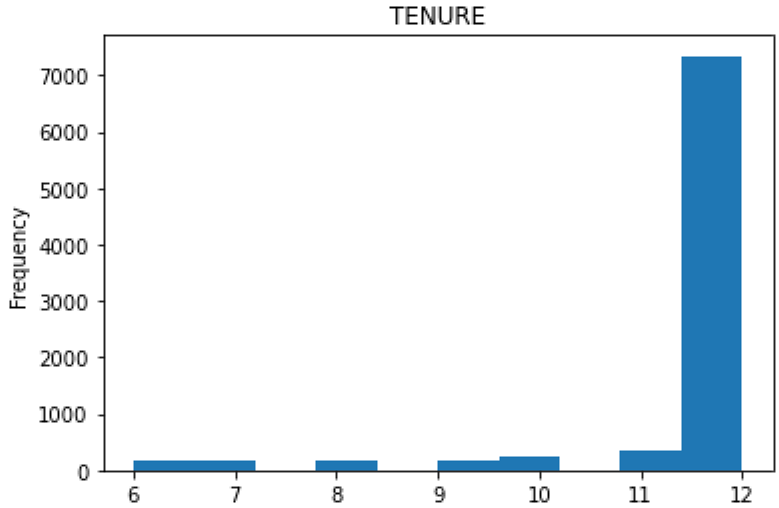

PURCHASES_TRX
Frequency
0
1000
2000
3000
4000
5000
6000
7000
8000
0
50
100
150
200
250
300
350


CREDIT_LIMIT
Frequency
0
1000
2000
3000
4000
0
5000
10000
15000
20000
25000
30000


PAYMENTS
Frequency
8000
7000
6000
5000
4000
3000
2000
1000
0
0
10000
20000
30000
40000
50000


MINIMUM_PAYMENTS
Frequency
8000
7000
6000
5000
4000
3000
2000
1000
0
0
10000
20000
30000
40000
50000
60000
70000
80000


PRC_FULL_PAYMENT
Frequency
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0



In [16]:

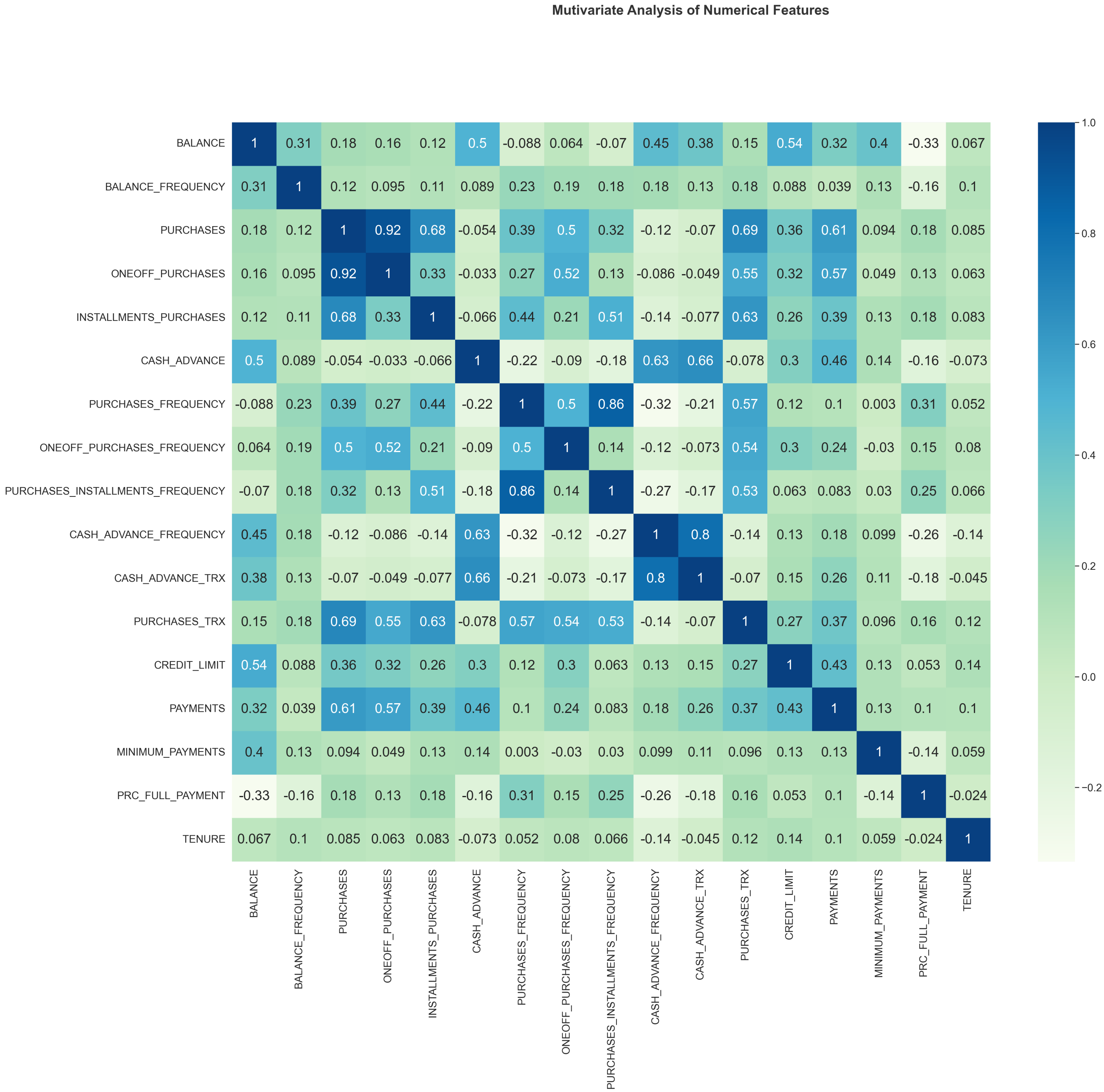
```
#multivariate analysis#
```

In [17]:

```
plt.figure(figsize=(25, 20),dpi=200)
plt.suptitle('Mutivariate Analysis of Numerical Features', fontsize=20, fontweight='bold',
correlation = df.corr()
sns.set(font_scale=1.4)
sns.heatmap(correlation,cmap='GnBu',annot=True,annot_kws={'size': 20})
```

Out[17]:

```
<AxesSubplot:>
```

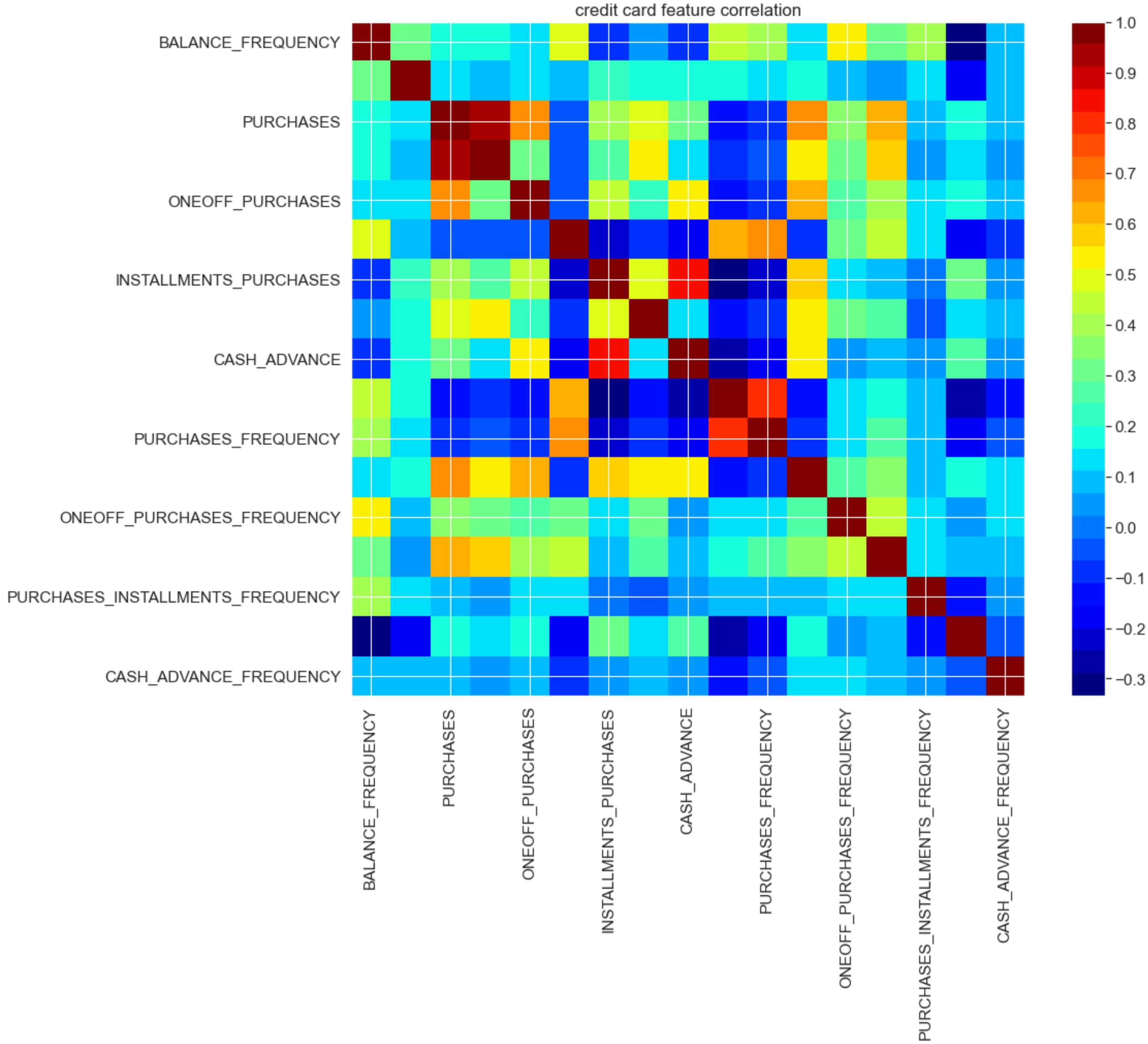


In [18]:

```
def correltion_mat(df):

    fig = plt.figure(figsize=(16,12))
    ax1 = fig.add_subplot(111)
    cmap = cm.get_cmap('jet',30)
    cax = ax1.imshow(df.corr(),interpolation="nearest",cmap=cmap)
    ax1.grid(True)
    plt.title("credit card feature correlation")
    labels=df.columns
    ax1.set_xticklabels(labels=labels,rotation=90)
    ax1.set_yticklabels(labels=labels)
    #colorbar
    fig.colorbar(cax,ticks=[0.1*i for i in range(-11,11)])
    plt.show();
```

In [19]:

```
correltion_mat(df)
```



In [20]:
```
important_col = ['BALANCE_FREQUENCY', 'PURCHASES', 'ONEOFF_PURCHASES','INSTALLMENTS_PURCHA
```

In [21]:
```
correlation
```

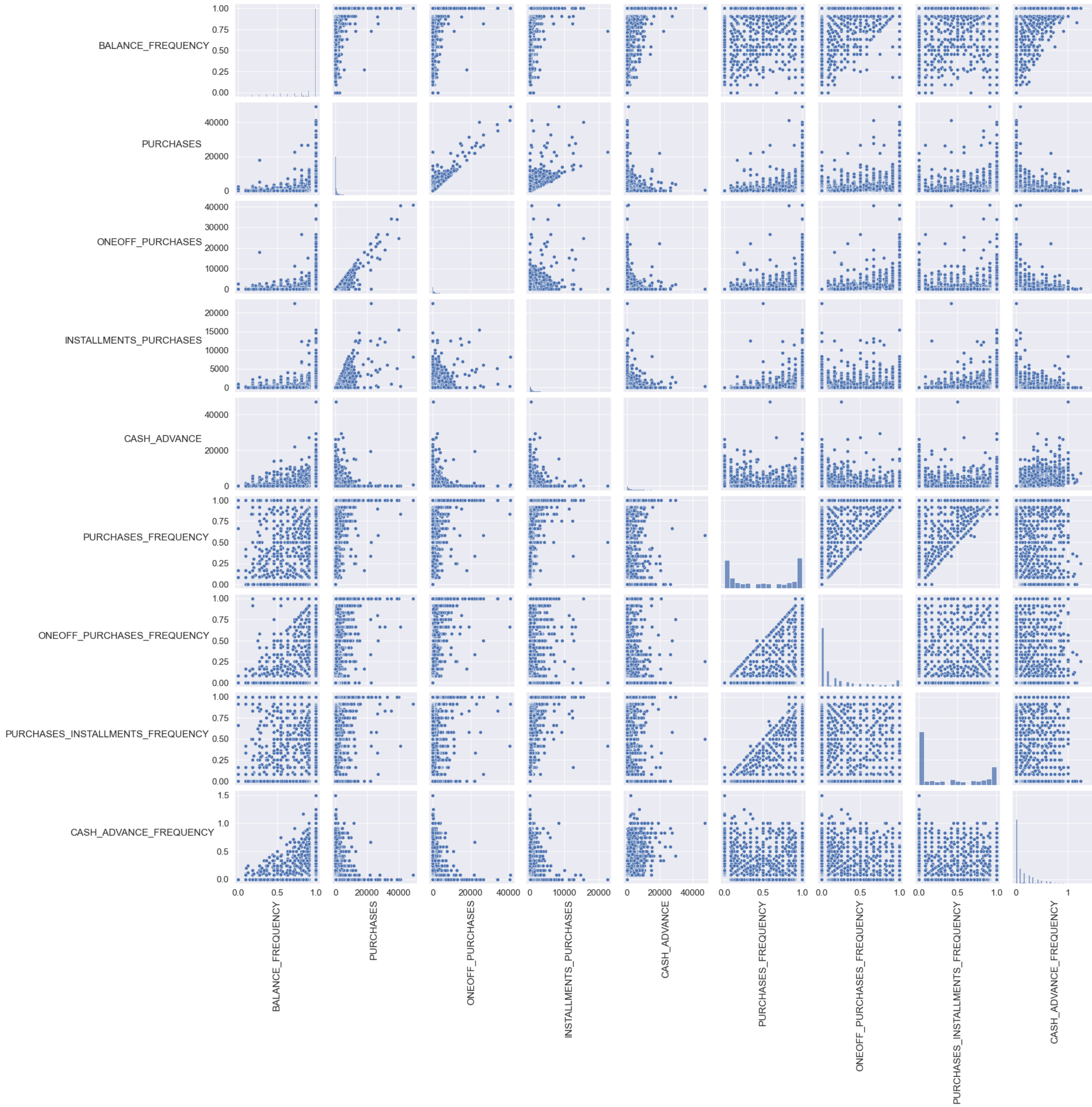
Out[21]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTA
BALANCE	1.000000	0.310140	0.176083	0.159985	
BALANCE_FREQUENCY	0.310140	1.000000	0.122635	0.095254	
PURCHASES	0.176083	0.122635	1.000000	0.916780	
ONEOFF_PURCHASES	0.159985	0.095254	0.916780	1.000000	
INSTALLMENTS_PURCHASES	0.122109	0.114739	0.679259	0.329650	
CASH_ADVANCE	0.495586	0.089036	-0.053760	-0.033244	
PURCHASES_FREQUENCY	-0.088459	0.228158	0.393000	0.265460	
ONEOFF_PURCHASES_FREQUENCY	0.063832	0.187467	0.497384	0.524514	
PURCHASES_INSTALLMENTS_FREQUENCY	-0.069582	0.184159	0.316025	0.128380	

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTA
CASH_ADVANCE_FREQUENCY	0.445307	0.181132	-0.124863	-0.086413	
CASH_ADVANCE_TRX	0.382388	0.133265	-0.070277	-0.048705	
PURCHASES_TRX	0.147887	0.183095	0.688732	0.545313	
CREDIT_LIMIT	0.535518	0.087682	0.358425	0.320613	
PAYMENTS	0.322830	0.039169	0.606782	0.570850	
MINIMUM_PAYMENTS	0.398669	0.132519	0.093842	0.048741	
PRC_FULL_PAYMENT	-0.333594	-0.156961	0.176447	0.129890	
TENURE	0.066987	0.104714	0.084545	0.063400	

In [22]:

```
g = sns.pairplot(df[important_col])
for ax in g.axes.flatten():
    # rotate x axis labels
    ax.set_xlabel(ax.get_xlabel(), rotation = 90)
    # rotate y axis labels
    ax.set_ylabel(ax.get_ylabel(), rotation = 0)
    # set y labels alignment
    ax.yaxis.get_label().set_horizontalalignment('right')
```



PCA analysis

In [23]:
```
#Normalizing th data for PCA analysis
from sklearn.preprocessing import StandardScaler
```

In [24]:
```
scaler=StandardScaler()
```

In [25]:
```
X = scaler.fit_transform(df)
```

In [26]:
```
dfx=pd.DataFrame(data=X,columns=df.columns)
```

In [27]:
```
dfx.describe()
```

Out[27]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_
count	8.636000e+03	8.636000e+03	8.636000e+03	8.636000e+03	8.636000e+03	8.6
mean	-2.506872e-18	-5.691243e-15	4.031476e-16	-5.819743e-15	2.643555e-15	-4.5
std	1.000058e+00	1.000058e+00	1.000058e+00	1.000058e+00	1.000058e+00	1.0
min	-7.641437e-01	-4.309583e+00	-4.732082e-01	-3.591603e-01	-4.588390e-01	-4.6
25%	-6.934691e-01	6.767893e-02	-4.531953e-01	-3.591603e-01	-4.588390e-01	-4.6
50%	-3.265978e-01	5.054046e-01	-2.999696e-01	-3.324445e-01	-3.554965e-01	-4.6
75%	2.405073e-01	5.054046e-01	5.562856e-02	-3.444604e-03	6.901931e-02	6.5
max	8.323708e+00	5.054046e-01	2.215714e+01	2.384284e+01	2.407255e+01	2.1

In [28]:

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

In [29]:

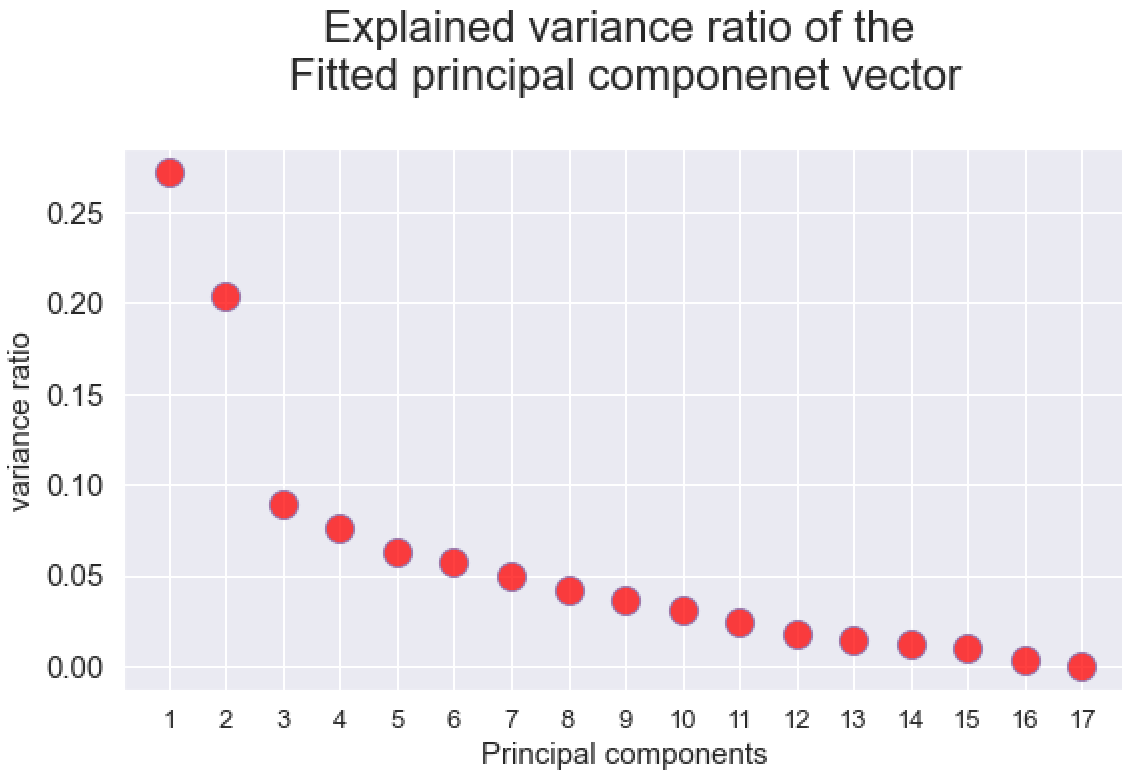
```
pca = PCA(n_components=None)
```

In [30]:

```
dfx_pca = pca.fit(dfx)
```

In [31]:

```
plt.figure(figsize=(9,5))
plt.scatter(x=[i+1 for i in range(len(dfx_pca.explained_variance_ratio_))],
            y=dfx_pca.explained_variance_ratio_,
            s=200,alpha=0.75,c='red',edgecolor='m')
plt.grid(True)
plt.title("Explained variance ratio of the \nFitted principal componenet vector\n",fontsiz
plt.xlabel("Principal components",fontsize=15)
plt.xticks([i+1 for i in range(len(dfx_pca.explained_variance_ratio_))],fontsize=13)
plt.yticks(fontsize=15)
plt.ylabel("variance ratio",fontsize=15)
plt.show()
```



In [32]:
```
print(dfx_pca.explained_variance_ratio_)
```

```
[2.72311770e-01 2.03743076e-01 8.91833372e-02 7.57360860e-02
 6.27661816e-02 5.71278613e-02 4.91618788e-02 4.21073427e-02
 3.68169669e-02 3.08150790e-02 2.36380362e-02 1.77453962e-02
 1.42671026e-02 1.17865972e-02 1.00809717e-02 2.71162672e-03
 6.90045972e-07]
```

In [33]:
```
#percentage of variance explained by each principal compoenet
for i, component in enumerate(dfx_pca.components_):
    print("{} component: {}% of initial variance".format(i + 1,
          round(100 * dfx_pca.explained_variance_ratio_[i], 2)))
    print(" + ".join("%.3f x %s" % (value, name)
                     for value, name in zip(component,
                                            df.columns)))
```

```
1 component: 27.23% of initial variance
0.092 x BALANCE + 0.110 x BALANCE_FREQUENCY + 0.412 x PURCHASES + 0.347 x ONEOFF_PURCHASES
+ 0.337 x INSTALLMENTS_PURCHASES + -0.031 x CASH_ADVANCE + 0.324 x PURCHASES_FREQUENCY +
0.295 x ONEOFF_PURCHASES_FREQUENCY + 0.277 x PURCHASES_INSTALLMENTS_FREQUENCY + -0.099 x C
ASH_ADVANCE_FREQUENCY + -0.057 x CASH_ADVANCE_TRX + 0.391 x PURCHASES_TRX + 0.210 x CREDIT
_LIMIT + 0.264 x PAYMENTS + 0.059 x MINIMUM_PAYMENTS + 0.131 x PRC_FULL_PAYMENT + 0.078 x
TENURE
2 component: 20.37% of initial variance
0.406 x BALANCE + 0.128 x BALANCE_FREQUENCY + 0.050 x PURCHASES + 0.070 x ONEOFF_PURCHASES
+ -0.011 x INSTALLMENTS_PURCHASES + 0.437 x CASH_ADVANCE + -0.187 x PURCHASES_FREQUENCY +
-0.015 x ONEOFF_PURCHASES_FREQUENCY + -0.174 x PURCHASES_INSTALLMENTS_FREQUENCY + 0.430 x
CASH_ADVANCE_FREQUENCY + 0.416 x CASH_ADVANCE_TRX + -0.012 x PURCHASES_TRX + 0.244 x CREDI
T_LIMIT + 0.264 x PAYMENTS + 0.170 x MINIMUM_PAYMENTS + -0.196 x PRC_FULL_PAYMENT + -0.005
x TENURE
3 component: 8.92% of initial variance
-0.174 x BALANCE + -0.459 x BALANCE_FREQUENCY + 0.243 x PURCHASES + 0.369 x ONEOFF_PURCHAS
ES + -0.104 x INSTALLMENTS_PURCHASES + -0.002 x CASH_ADVANCE + -0.356 x PURCHASES_FREQUENC
Y + 0.105 x ONEOFF_PURCHASES_FREQUENCY + -0.450 x PURCHASES_INSTALLMENTS_FREQUENCY + -0.08
8 x CASH_ADVANCE_FREQUENCY + -0.087 x CASH_ADVANCE_TRX + -0.080 x PURCHASES_TRX + 0.095 x
CREDIT_LIMIT + 0.288 x PAYMENTS + -0.249 x MINIMUM_PAYMENTS + 0.184 x PRC_FULL_PAYMENT + -
0.066 x TENURE
4 component: 7.57% of initial variance
```

```
0.259 x BALANCE + 0.159 x BALANCE_FREQUENCY + 0.064 x PURCHASES + 0.123 x ONEOFF_PURCHASES
+ -0.075 x INSTALLMENTS_PURCHASES + -0.266 x CASH_ADVANCE + -0.222 x PURCHASES_FREQUENCY +
0.055 x ONEOFF_PURCHASES_FREQUENCY + -0.265 x PURCHASES_INSTALLMENTS_FREQUENCY + -0.267 x
CASH_ADVANCE_FREQUENCY + -0.333 x CASH_ADVANCE_TRX + -0.024 x PURCHASES_TRX + 0.123 x CRED
IT_LIMIT + -0.098 x PAYMENTS + 0.352 x MINIMUM_PAYMENTS + -0.418 x PRC_FULL_PAYMENT + 0.42
8 x TENURE
5 component: 6.28% of initial variance
0.076 x BALANCE + -0.451 x BALANCE_FREQUENCY + -0.010 x PURCHASES + -0.197 x ONEOFF_PURCHA
SES + 0.337 x INSTALLMENTS_PURCHASES + 0.099 x CASH_ADVANCE + -0.089 x PURCHASES_FREQUENCY
+ -0.522 x ONEOFF_PURCHASES_FREQUENCY + 0.175 x PURCHASES_INSTALLMENTS_FREQUENCY + -0.160
x CASH_ADVANCE_FREQUENCY + -0.090 x CASH_ADVANCE_TRX + -0.053 x PURCHASES_TRX + 0.132 x CR
EDIT_LIMIT + 0.189 x PAYMENTS + 0.417 x MINIMUM_PAYMENTS + 0.201 x PRC_FULL_PAYMENT + 0.11
8 x TENURE
6 component: 5.71% of initial variance
0.036 x BALANCE + -0.015 x BALANCE_FREQUENCY + 0.196 x PURCHASES + 0.173 x ONEOFF_PURCHASE
S + 0.145 x INSTALLMENTS_PURCHASES + -0.133 x CASH_ADVANCE + -0.086 x PURCHASES_FREQUENCY
+ -0.097 x ONEOFF_PURCHASES_FREQUENCY + -0.047 x PURCHASES_INSTALLMENTS_FREQUENCY + 0.032
x CASH_ADVANCE_FREQUENCY + -0.090 x CASH_ADVANCE_TRX + 0.078 x PURCHASES_TRX + -0.313 x CR
EDIT_LIMIT + -0.066 x PAYMENTS + 0.340 x MINIMUM_PAYMENTS + -0.289 x PRC_FULL_PAYMENT + -
0.746 x TENURE
7 component: 4.92% of initial variance
-0.263 x BALANCE + 0.099 x BALANCE_FREQUENCY + 0.201 x PURCHASES + 0.113 x ONEOFF_PURCHASE
S + 0.269 x INSTALLMENTS_PURCHASES + -0.039 x CASH_ADVANCE + -0.158 x PURCHASES_FREQUENCY
+ -0.306 x ONEOFF_PURCHASES_FREQUENCY + 0.043 x PURCHASES_INSTALLMENTS_FREQUENCY + 0.137 x
CASH_ADVANCE_FREQUENCY + 0.197 x CASH_ADVANCE_TRX + 0.104 x PURCHASES_TRX + -0.544 x CREDI
T_LIMIT + 0.169 x PAYMENTS + -0.204 x MINIMUM_PAYMENTS + -0.280 x PRC_FULL_PAYMENT + 0.401
x TENURE
8 component: 4.21% of initial variance
-0.200 x BALANCE + 0.128 x BALANCE_FREQUENCY + -0.005 x PURCHASES + 0.123 x ONEOFF_PURCHAS
ES + -0.238 x INSTALLMENTS_PURCHASES + -0.005 x CASH_ADVANCE + 0.026 x PURCHASES_FREQUENCY
+ 0.200 x ONEOFF_PURCHASES_FREQUENCY + -0.129 x PURCHASES_INSTALLMENTS_FREQUENCY + 0.077 x
CASH_ADVANCE_FREQUENCY + 0.180 x CASH_ADVANCE_TRX + -0.045 x PURCHASES_TRX + -0.367 x CRED
IT_LIMIT + 0.048 x PAYMENTS + 0.613 x MINIMUM_PAYMENTS + 0.482 x PRC_FULL_PAYMENT + 0.169
x TENURE
9 component: 3.68% of initial variance
0.062 x BALANCE + 0.671 x BALANCE_FREQUENCY + 0.101 x PURCHASES + 0.069 x ONEOFF_PURCHASES
+ 0.112 x INSTALLMENTS_PURCHASES + -0.019 x CASH_ADVANCE + -0.191 x PURCHASES_FREQUENCY +
-0.362 x ONEOFF_PURCHASES_FREQUENCY + -0.082 x PURCHASES_INSTALLMENTS_FREQUENCY + -0.087 x
CASH_ADVANCE_FREQUENCY + -0.215 x CASH_ADVANCE_TRX + -0.255 x PURCHASES_TRX + 0.094 x CRED
IT_LIMIT + 0.136 x PAYMENTS + -0.148 x MINIMUM_PAYMENTS + 0.393 x PRC_FULL_PAYMENT + -0.14
4 x TENURE
10 component: 3.08% of initial variance
0.045 x BALANCE + -0.027 x BALANCE_FREQUENCY + 0.059 x PURCHASES + -0.165 x ONEOFF_PURCHAS
ES + 0.444 x INSTALLMENTS_PURCHASES + -0.374 x CASH_ADVANCE + -0.258 x PURCHASES_FREQUENCY
+ 0.089 x ONEOFF_PURCHASES_FREQUENCY + -0.256 x PURCHASES_INSTALLMENTS_FREQUENCY + 0.290 x
CASH_ADVANCE_FREQUENCY + 0.208 x CASH_ADVANCE_TRX + 0.230 x PURCHASES_TRX + 0.161 x CREDIT
_LIMIT + -0.460 x PAYMENTS + -0.017 x MINIMUM_PAYMENTS + 0.266 x PRC_FULL_PAYMENT + 0.041
x TENURE
11 component: 2.36% of initial variance
0.151 x BALANCE + -0.139 x BALANCE_FREQUENCY + 0.196 x PURCHASES + 0.446 x ONEOFF_PURCHASE
S + -0.356 x INSTALLMENTS_PURCHASES + -0.353 x CASH_ADVANCE + 0.126 x PURCHASES_FREQUENCY
+ -0.370 x ONEOFF_PURCHASES_FREQUENCY + 0.296 x PURCHASES_INSTALLMENTS_FREQUENCY + 0.213 x
CASH_ADVANCE_FREQUENCY + 0.209 x CASH_ADVANCE_TRX + -0.203 x PURCHASES_TRX + 0.152 x CREDI
T_LIMIT + -0.260 x PAYMENTS + -0.022 x MINIMUM_PAYMENTS + 0.050 x PRC_FULL_PAYMENT + 0.066
x TENURE
12 component: 1.77% of initial variance
-0.476 x BALANCE + 0.067 x BALANCE_FREQUENCY + 0.079 x PURCHASES + -0.049 x ONEOFF_PURCHAS
ES + 0.277 x INSTALLMENTS_PURCHASES + -0.174 x CASH_ADVANCE + 0.161 x PURCHASES_FREQUENCY
+ 0.166 x ONEOFF_PURCHASES_FREQUENCY + -0.017 x PURCHASES_INSTALLMENTS_FREQUENCY + 0.038 x
CASH_ADVANCE_FREQUENCY + 0.204 x CASH_ADVANCE_TRX + -0.594 x PURCHASES_TRX + 0.321 x CREDI
T_LIMIT + 0.118 x PAYMENTS + 0.160 x MINIMUM_PAYMENTS + -0.246 x PRC_FULL_PAYMENT + -0.031
x TENURE
13 component: 1.43% of initial variance
0.538 x BALANCE + -0.169 x BALANCE_FREQUENCY + 0.109 x PURCHASES + -0.011 x ONEOFF_PURCHAS
ES + 0.277 x INSTALLMENTS_PURCHASES + 0.011 x CASH_ADVANCE + 0.194 x PURCHASES_FREQUENCY +
0.248 x ONEOFF_PURCHASES_FREQUENCY + -0.041 x PURCHASES_INSTALLMENTS_FREQUENCY + 0.043 x C
```

```
ASH_ADVANCE_FREQUENCY + -0.094 x CASH_ADVANCE_TRX + -0.530 x PURCHASES_TRX + -0.402 x CRED
IT_LIMIT + -0.042 x PAYMENTS + -0.141 x MINIMUM_PAYMENTS + 0.113 x PRC_FULL_PAYMENT + 0.07
7 x TENURE
14 component: 1.18% of initial variance
0.143 x BALANCE + -0.023 x BALANCE_FREQUENCY + -0.225 x PURCHASES + -0.223 x ONEOFF_PURCHA
SES + -0.121 x INSTALLMENTS_PURCHASES + -0.597 x CASH_ADVANCE + 0.010 x PURCHASES_FREQUENC
Y + 0.044 x ONEOFF_PURCHASES_FREQUENCY + 0.044 x PURCHASES_INSTALLMENTS_FREQUENCY + 0.338
x CASH_ADVANCE_FREQUENCY + -0.118 x CASH_ADVANCE_TRX + 0.080 x PURCHASES_TRX + -0.030 x CR
EDIT_LIMIT + 0.604 x PAYMENTS + -0.024 x MINIMUM_PAYMENTS + 0.009 x PRC_FULL_PAYMENT + -0.
048 x TENURE
15 component: 1.01% of initial variance
0.219 x BALANCE + 0.044 x BALANCE_FREQUENCY + -0.063 x PURCHASES + -0.068 x ONEOFF_PURCHAS
ES + -0.023 x INSTALLMENTS_PURCHASES + -0.238 x CASH_ADVANCE + -0.024 x PURCHASES_FREQUENC
Y + -0.015 x ONEOFF_PURCHASES_FREQUENCY + -0.067 x PURCHASES_INSTALLMENTS_FREQUENCY + -0.6
47 x CASH_ADVANCE_FREQUENCY + 0.649 x CASH_ADVANCE_TRX + 0.038 x PURCHASES_TRX + -0.054 x
CREDIT_LIMIT + 0.138 x PAYMENTS + -0.072 x MINIMUM_PAYMENTS + 0.011 x PRC_FULL_PAYMENT + -
0.104 x TENURE
16 component: 0.27% of initial variance
-0.006 x BALANCE + -0.009 x BALANCE_FREQUENCY + 0.001 x PURCHASES + -0.005 x ONEOFF_PURCHA
SES + 0.014 x INSTALLMENTS_PURCHASES + -0.008 x CASH_ADVANCE + 0.679 x PURCHASES_FREQUENCY
+ -0.342 x ONEOFF_PURCHASES_FREQUENCY + -0.633 x PURCHASES_INSTALLMENTS_FREQUENCY + 0.042
x CASH_ADVANCE_FREQUENCY + -0.014 x CASH_ADVANCE_TRX + 0.135 x PURCHASES_TRX + 0.019 x CRE
DIT_LIMIT + 0.011 x PAYMENTS + -0.015 x MINIMUM_PAYMENTS + -0.021 x PRC_FULL_PAYMENT + 0.0
20 x TENURE
17 component: 0.0% of initial variance
0.000 x BALANCE + 0.000 x BALANCE_FREQUENCY + -0.749 x PURCHASES + 0.582 x ONEOFF_PURCHASE
S + 0.317 x INSTALLMENTS_PURCHASES + 0.000 x CASH_ADVANCE + -0.000 x PURCHASES_FREQUENCY +
0.000 x ONEOFF_PURCHASES_FREQUENCY + 0.000 x PURCHASES_INSTALLMENTS_FREQUENCY + -0.000 x C
ASH_ADVANCE_FREQUENCY + 0.000 x CASH_ADVANCE_TRX + -0.000 x PURCHASES_TRX + -0.000 x CREDI
T_LIMIT + -0.000 x PAYMENTS + 0.000 x MINIMUM_PAYMENTS + 0.000 x PRC_FULL_PAYMENT + -0.000
x TENURE
```

In [34]:
```
dfx_trans=pca.transform(dfx)
```

In [35]:
```
dfx_trans=pd.DataFrame(data=dfx_trans)
```

In [36]:
```
dfx_trans.head()
```

Out[36]:

	0	1	2	3	4	5	6	7	8	9	
0	-1.696395	-1.122584	0.491562	0.719521	0.079830	0.118234	0.808993	-0.093970	-0.016190	-0.082402	-0.216
1	-1.215681	2.435638	0.694658	-0.098843	0.803019	-0.917777	-0.322969	-0.045119	0.754617	-0.748468	-0.878
2	0.935853	-0.385202	-0.025953	1.293844	-1.987285	-0.682139	-1.624721	0.073401	-0.837066	-0.034854	-0.746
3	-1.614638	-0.724586	0.272358	1.086116	-0.427814	0.082982	0.687001	0.063548	0.566940	-0.083532	-0.466
4	0.223701	-0.783610	-1.184434	0.721353	0.801243	0.525879	0.788893	-0.089942	0.365857	-0.192647	-0.194

In [37]:
```
# taking 5 principal components for the clustering model
pca5 = PCA(n_components=5)
pca5.fit(dfx)
dfx_trans5=pca5.transform(dfx)
```

K-means clustering function

In [38]:
```
from sklearn.cluster import KMeans
```

```
from sklearn import metrics
```

In [39]:
```
def Kmeans_algo(dataset,n):
    c_k_mean = KMeans(n_clusters=n,init='k-means++',max_iter=300,random_state=42,algorithm
    c_k_mean.fit(dataset)

    #Creating dataframe to store centroids
    centroids = c_k_mean.cluster_centers_
    centroids_df = pd.DataFrame(centroids,columns=['X','Y'])

    #add cluster label to each dta point
    label = c_k_mean.labels_
    df["label"]=label

    #evaluation metrics for clustering - inertia and silhouette score

    inertia = c_k_mean.inertia_
    silhouette_score= metrics.silhouette_score(dataset,label)

    return label, centroids_df, inertia, silhouette_score
```

visulaizing data with different features got by correlation matrix

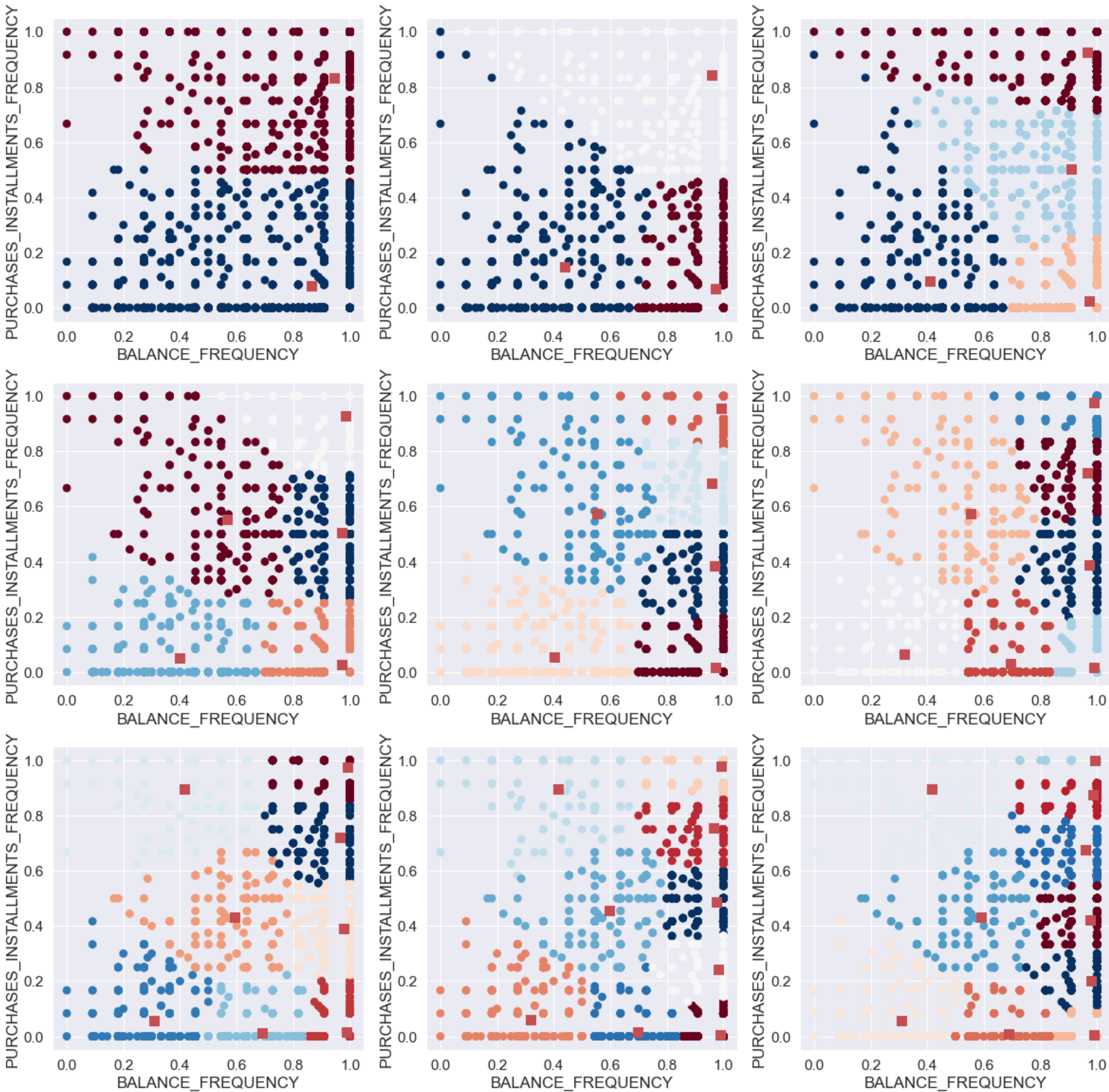
In [40]:
```
#comparing differnet aspects of data wit each other.
X1=df[['BALANCE_FREQUENCY', 'PURCHASES']].values
X2=df[['PURCHASES', 'ONEOFF_PURCHASES']].values
X3=df[['ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES']].values
X4=df[['INSTALLMENTS_PURCHASES', 'CASH_ADVANCE']].values
X5=df[['CASH_ADVANCE', 'PURCHASES_FREQUENCY']].values
X6=df[['PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY']].values
X7=df[['ONEOFF_PURCHASES_FREQUENCY', 'PURCHASES_INSTALLMENTS_FREQUENCY']].values
X8=df[['PURCHASES_INSTALLMENTS_FREQUENCY', 'CASH_ADVANCE_FREQUENCY']].values
X9=df[['CREDIT_LIMIT', 'TENURE','BALANCE']].values
X10=df[['BALANCE_FREQUENCY', 'PURCHASES_INSTALLMENTS_FREQUENCY']].values
```

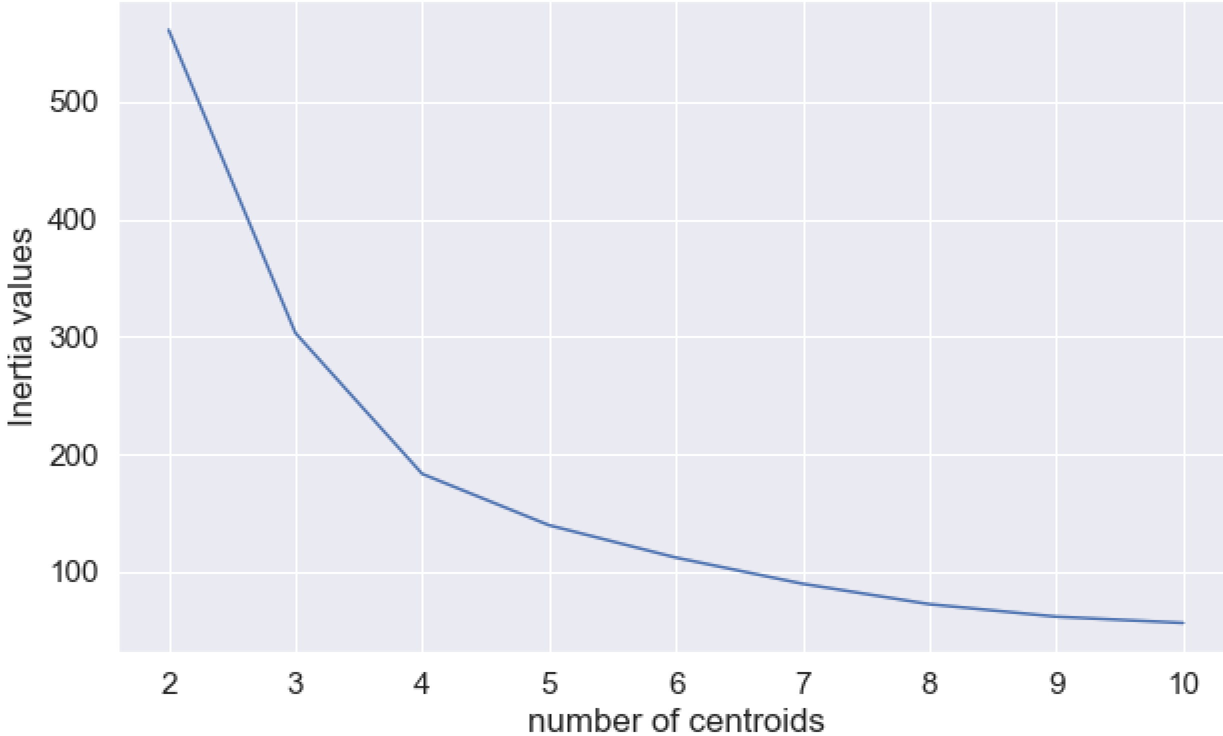
In [41]:
```
# Blance frequency and purchases
X10_inertia_values=[]
X10_silhouette_scores=[]
fig10=plt.figure(figsize=(20,20))
for i in range (2,11):
    X10_label, X10_centriods,X10_inertia, X10_silhoutte = Kmeans_algo(X10,i)
    X10_inertia_values.append(X10_inertia)
    X10_silhouette_scores.append(X10_silhoutte)
    centroids_df = pd.DataFrame(X10_centriods,columns = ['X','Y'])

    sub = fig10.add_subplot(330+i-1)
    sub.scatter(df['BALANCE_FREQUENCY'],df['PURCHASES_INSTALLMENTS_FREQUENCY'],s=60,c=df['
    sub.scatter(centroids_df['X'],centroids_df["Y"],s=90,marker=",",color='r')
    sub.set_xlabel('BALANCE_FREQUENCY')
    sub.set_ylabel('PURCHASES_INSTALLMENTS_FREQUENCY')

#Plot inertia values against number of clusters
plt.figure(figsize=(10,6))
plt.plot(np.arange(2,11),X10_inertia_values)
plt.xlabel("number of centroids")
plt.ylabel("Inertia values")

#Plot silhouette_score values against number of clusters
```

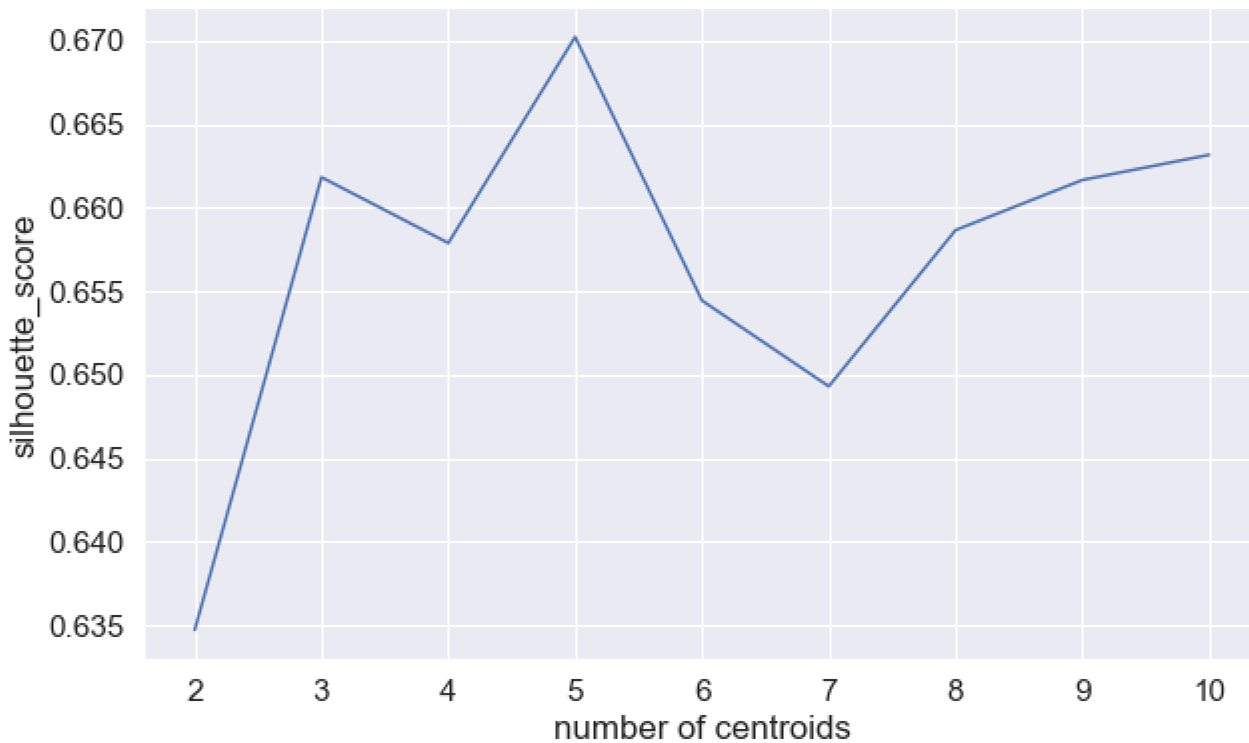
```python
plt.figure(figsize=(10,6))
plt.plot(np.arange(2,11),X10_silhouette_scores)
plt.xlabel("number of centroids")
plt.ylabel("silhouette_score")
```

Out[41]:

```
Text(0, 0.5, 'silhouette_score')
```





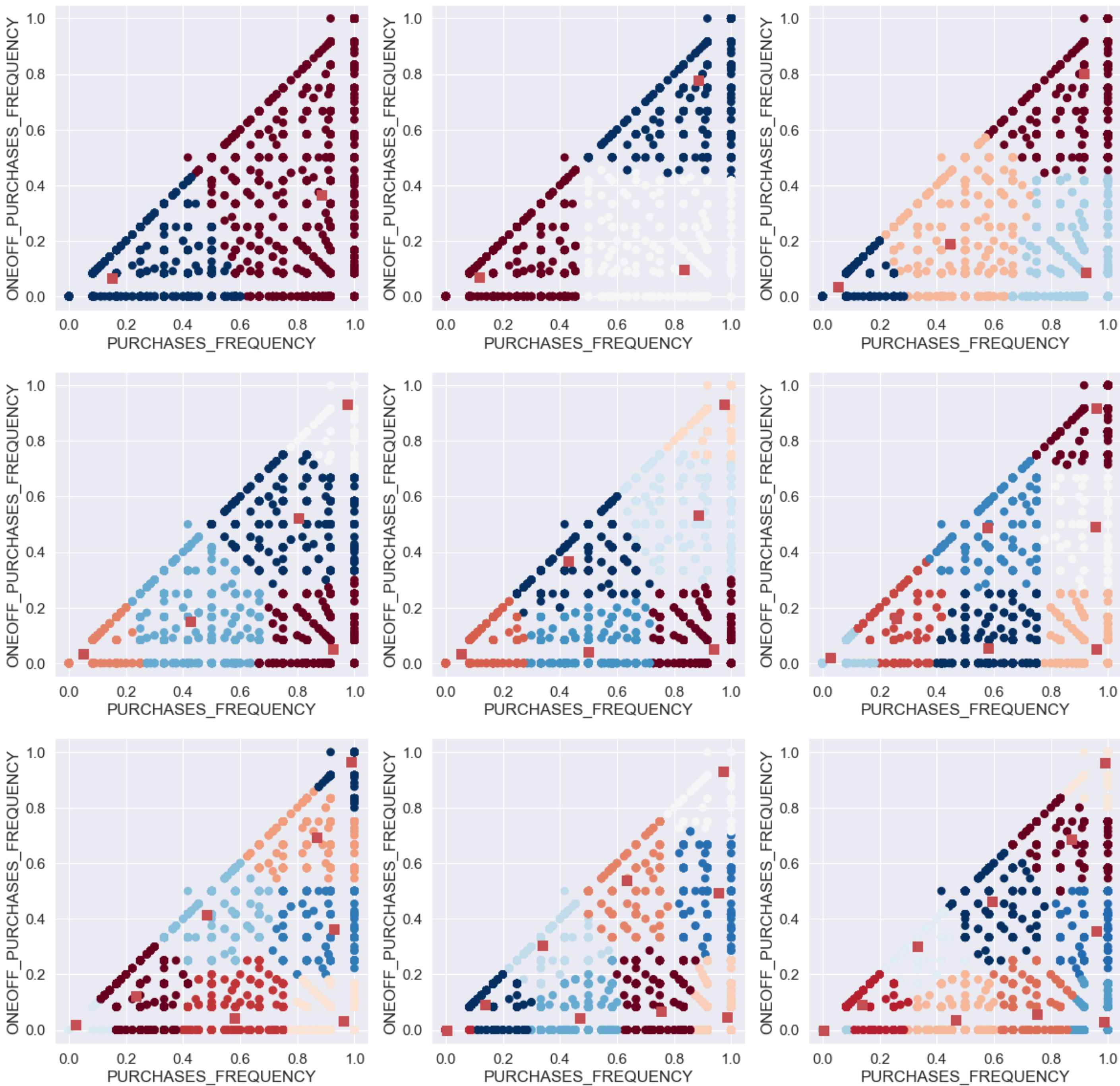


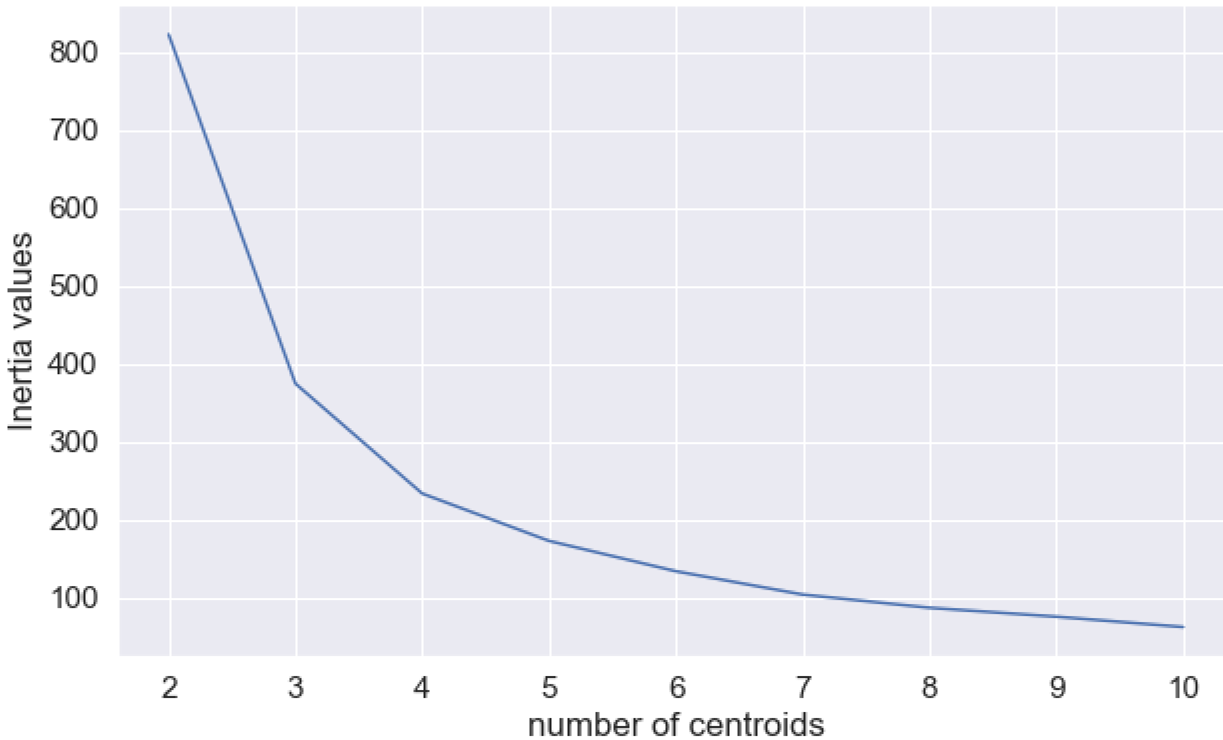
In [42]:

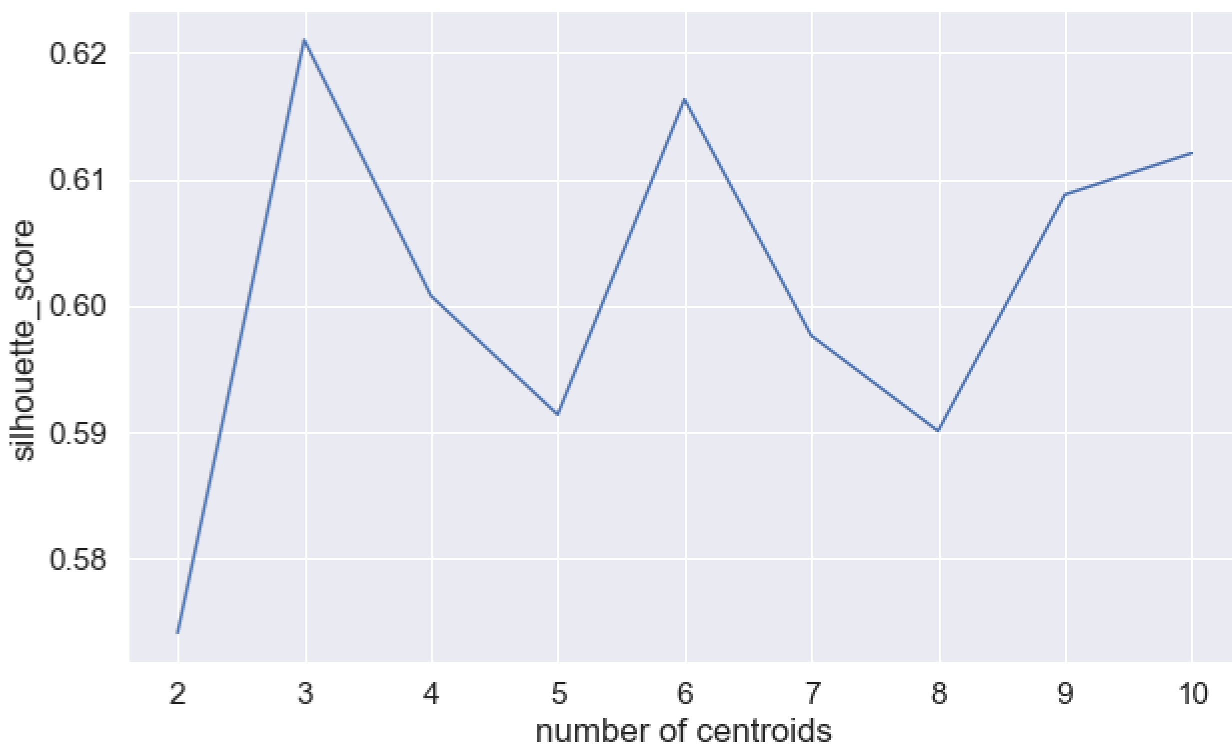
```
# Blance frequency and purchases
X6_inertia_values=[]
X6_silhouette_scores=[]
fig6=plt.figure(figsize=(20,20))
for i in range (2,11):
    X6_label, X6_centriods,X6_inertia, X6_silhoutte = Kmeans_algo(X6,i)
    X6_inertia_values.append(X6_inertia)
    X6_silhouette_scores.append(X6_silhoutte)
    centroids_df = pd.DataFrame(X6_centriods,columns = ['X','Y'])

    sub = fig6.add_subplot(330+i-1)
    sub.scatter(df['PURCHASES_FREQUENCY'],df['ONEOFF_PURCHASES_FREQUENCY'],s=60,c=df['labe
    sub.scatter(centroids_df['X'],centroids_df["Y"],s=90,marker=",",color='r')
    sub.set_xlabel('PURCHASES_FREQUENCY')
    sub.set_ylabel('ONEOFF_PURCHASES_FREQUENCY')
```

```
#Plot inertia values against number of clusters
plt.figure(figsize=(10,6))
plt.plot(np.arange(2,11),X6_inertia_values)
plt.xlabel("number of centroids")
plt.ylabel("Inertia values")

#Plot silhouette_score values against number of clusters
plt.figure(figsize=(10,6))
plt.plot(np.arange(2,11),X6_silhouette_scores)
plt.xlabel("number of centroids")
plt.ylabel("silhouette_score")
```

Out[42]: Text(0, 0.5, 'silhouette_score')







we will get same value for k if e check for all features, so taking k as 5

In [43]:
```
# K=5 is the optimal number of cluster

c_k_mean5 = KMeans(n_clusters=5,init='k-means++',max_iter=300,random_state=42,algorithm="e
c_k_mean5.fit(df)
```

Out[43]:
```
KMeans(algorithm='elkan', n_clusters=5, random_state=42)
```

In [44]:
```
df['k_m_5']=c_k_mean5.predict(df)
```

In [45]:
```
profile = df.groupby('k_m_5').mean().T
```

In [46]:
```python
round(profile)
```

Out[46]:

k_m_5	0	1	2	3	4
BALANCE	818.0	5533.0	4058.0	4836.0	1665.0
BALANCE_FREQUENCY	1.0	1.0	1.0	1.0	1.0
PURCHASES	500.0	1526.0	1027.0	11819.0	1482.0
ONEOFF_PURCHASES	240.0	937.0	118.0	8432.0	923.0
INSTALLMENTS_PURCHASES	260.0	589.0	909.0	3387.0	559.0
CASH_ADVANCE	497.0	3885.0	923.0	5250.0	830.0
PURCHASES_FREQUENCY	0.0	0.0	0.0	1.0	1.0
ONEOFF_PURCHASES_FREQUENCY	0.0	0.0	0.0	1.0	0.0
PURCHASES_INSTALLMENTS_FREQUENCY	0.0	0.0	0.0	1.0	0.0
CASH_ADVANCE_FREQUENCY	0.0	0.0	0.0	0.0	0.0
CASH_ADVANCE_TRX	2.0	10.0	3.0	9.0	3.0
PURCHASES_TRX	9.0	21.0	19.0	88.0	21.0
CREDIT_LIMIT	2185.0	10950.0	4268.0	12718.0	6858.0
PAYMENTS	916.0	4068.0	1625.0	19034.0	2046.0
MINIMUM_PAYMENTS	517.0	1932.0	22760.0	2471.0	645.0
PRC_FULL_PAYMENT	0.0	0.0	0.0	0.0	0.0
TENURE	11.0	12.0	12.0	12.0	12.0
label	5.0	5.0	5.0	4.0	4.0

In [47]:
```python
# Clustering with respect to PCA
c_k_mean_pca = KMeans(n_clusters=5,init='k-means++',max_iter=300,random_state=42,algorithm
c_k_mean_pca.fit(dfx_trans5)
```

Out[47]:
```
KMeans(algorithm='elkan', n_clusters=5, random_state=42)
```

In [48]:
```python
pca_trans = c_k_mean_pca.transform(dfx_trans5)
```

In [49]:
```python
#creating a new dataframe with original features and adding PCA scores and clusters assig
df_segment_pca = pd.concat([df.reset_index(drop=True),pd.DataFrame(pca_trans)],axis=1)
```

In [50]:
```python
# creating different cloumns for cluster
df_segment_pca.columns.values[-5:] = ['component 1','component 2','component 3','component
```

In [51]:
```python
# Adding cluster labels to te dataframe
df_segment_pca["segmented K-means PCA"] = c_k_mean_pca.labels_
```

In [52]:
```python
df_segment_pca.head()
```

Out[52]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVAN
0	40.900749	0.818182	95.40	0.00	95.40	0.000
1	3202.467416	0.909091	0.00	0.00	0.00	6442.945
2	2495.148862	1.000000	773.17	773.17	0.00	0.000
3	817.714335	1.000000	16.00	16.00	0.00	0.000
4	1809.828751	1.000000	1333.28	0.00	1333.28	0.000

5 rows × 25 columns

In [53]:

```
#checking newly added column
df_segment_pca['segmented K-means PCA'].unique()
```

Out[53]:

```
array([3, 0, 1, 2, 4])
```

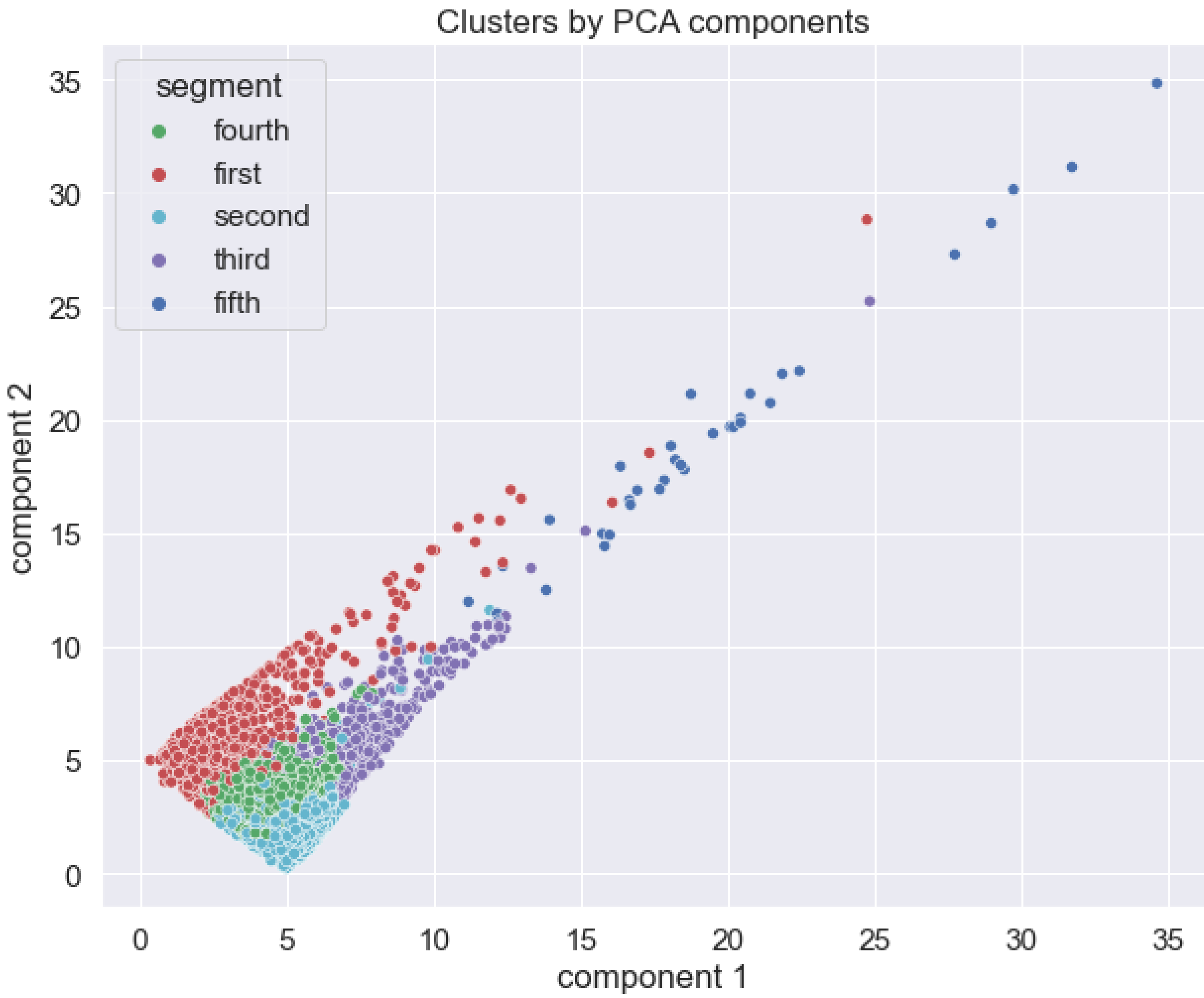
In [54]:

```
# mapping compoenets for visualization
df_segment_pca['segment'] = df_segment_pca['segmented K-means PCA'].map({0:'first',1:'seco
```

Visualization with PCA

In [55]:

```
x_axis = df_segment_pca['component 1']
y_axis = df_segment_pca['component 2']
plt.figure(figsize=(10,8))
sns.scatterplot(x_axis, y_axis,hue=df_segment_pca['segment'],palette=['g','r','c','m','b']
plt.title('Clusters by PCA components')
plt.show();
```


Clusters by PCA components
segment
fourth
first
second
third
fifth
component 2
component 1
0
5
10
15
20
25
30
35

```
In [ ]:
```