**Project - Credit Card Segmentation**

**Name: Jaydi Nithin Reddy**

**www.edwisor.com**

Bank wants to sell Credit Card depending on the behaviour of its customers

Business Case Overview: This case requires trainees to develop a customer segmentation to define marketing strategy. The sample dataset summarizes the usage behavior of about 9000 active credit card holders during the last 6 months. The file is at a customer level with 18 behavioral variables.

### **What is Customer Segmentation?**

Customer Segmentation is the process of division of customer base into several groups of individuals that share a similarity in different ways that are relevant to marketing such as gender, age, interests, and miscellaneous spending habits.

Companies that deploy customer segmentation are under the notion that every customer has different requirements and require a specific marketing effort to address them appropriately. Companies aim to gain a deeper approach of the customer they are targeting. Therefore, their aim has to be specific and should be tailored to address the requirements of each and every individual customer. Furthermore, through the data collected, companies can gain a deeper understanding of customer preferences as well as the requirements for discovering valuable segments that would reap them maximum profit. This way, they can strategize their marketing techniques more efficiently and minimize the possibility of risk to their investment.

The technique of customer segmentation is dependent on several key differentiators that divide customers into groups to be targeted. Data related to demographics, geography, economic status as well as behavioral patterns play a crucial role in determining the company direction towards addressing the various segments.

Missing Value Analysis:

The Missing Value Analysis procedure performs three primary functions:

* Describes the pattern of missing data. Where are the missing values located? How extensive are they? Do pairs of variables tend to have values missing in multiple cases? Are data values extreme? Are values missing randomly?
* Estimates means, standard deviations, covariances, and correlations for different missing value methods: listwise, pairwise, regression, or EM (expectation-maximization). The pairwise method also displays counts of pairwise complete cases.
* Fills in (imputes) missing values with estimated values using regression or EM methods; however, multiple imputation is generally considered to provide more accurate results.

Missing value analysis helps address several concerns caused by incomplete data. If cases with missing values are systematically different from cases without missing values, the results can be misleading. Also, missing data may reduce the precision of calculated statistics because there is less information than originally planned. Another concern is that the assumptions behind many statistical procedures are based on complete cases, and missing values can complicate the theory required.

Outlier Analysis:

An outlier is an element of a data set that distinctly stands out from the rest of the data. In other words, outliers are those data points that lie outside the overall pattern of distribution.

The easiest way to detect outliers is to create a graph. Plots such as Box plots, Scatterplots and Histograms can help to detect outliers. Alternatively, we can use mean and standard deviation to list out the outliers. Interquartile Range and Quartiles can also be used to detect outliers.

Outlier data points can represent either a) items that are so far outside the norm that they need not be considered or b) the illustration of a very unique and singular category or variable that is worth exploring either to capitalize on a niche or find an area where an organization can offer a unique focus.

When considering the use of Outlier analysis, a business should first think about why they want to find the outliers and what they will do with that data. That focus will help the business to select the right method of analysis, graphing or plotting to reveal the results they need to see and understand.

Data Cleaning:

Data cleaning involve different techniques based on the problem and the data type. Different methods can be applied with each has its own trade-offs.

Overall, incorrect data is either removed, corrected, or imputed.

**Duplicates:**

Duplicates are data points that are repeated in your dataset.

It often happens when for example

* Data are combined from different sources
* The user may hit submit button twice thinking the form wasn’t actually submitted.
* A request to online booking was submitted twice correcting wrong information that was entered accidentally in the first time.

A common symptom is when two users have the same identity number. Or, the same article was scrapped twice.

And therefore, they simply should be removed.

Factor Analysis:

**Factor analysis** is a technique that is used to reduce a large number of variables into fewer numbers of factors.  This technique extracts maximum common variance from all variables and puts them into a common score.  As an index of all variables, we can use this score for further analysis.  Factor analysis is part of general linear model (GLM) and this method also assumes several assumptions: there is linear relationship, there is no multicollinearity, it includes relevant variables into analysis, and there is true correlation between variables and factors.  Several methods are available, but principle component analysis is used most commonly.

**Data Standardization:**

Data Standardization is a data processing workflow that converts the structure of disparate datasets into a Common Data Format. As part of the Data Preparation field, Data Standardization deals with the transformation of datasets after the data is pulled from source systems and before it's loaded into target systems. Because of that, Data Standardization can also be thought of as the transformation rules engine in Data Exchange operations.

Data Standardization enables the data consumer to analyse and use data in a consistent manner. Typically, when data is created and stored in the source system, it's structured in a particular way that is often unknown to the data consumer. Moreover, datasets that might be semantically related may be stored and represented differently, thereby making it difficult for a data consumer to aggregate or compare the datasets.

**K-Means Clustering:**

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.

A cluster refers to a collection of data points aggregated together because of certain similarities. You’ll define a target number k, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the centre of the cluster. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

The ‘means’ in the K-means refers to averaging of the data; that is, finding the centroid.

**Profiling:**

Dataprofiling is the process of examining the data available from an existing information source (e.g. a database or a file) and collecting statistics or informative summaries about that data.

#### **Problems need to address:**

Advanced data preparation: Build an ‘enriched’ customer profile by deriving “intelligent” KPIs such as:

* Monthly average purchase and cash advance amount
* Purchases by type (one-off, installments)
* Average amount per purchase and cash advance transaction,
* Limit usage (balance to credit limit ratio),
* Payments to minimum payments ratio etc.

#### **Advanced reporting:**

* Use the derived KPIs to gain insight on the customer profiles.
* Identification of the relationships/ affinities between services.
* Clustering: Apply a data reduction technique factor analysis for variable reduction technique and a clustering algorithm to reveal the behavioural segments of credit card holders
* Identify cluster characterisitics of the cluster using detailed profiling.
* Provide the strategic insights and implementation of strategies for given set of cluster characteristics

#### **DATA DICTIONARY:**

* CUST\_ID: Credit card holder ID
* BALANCE: Monthly average balance (based on daily balance averages)
* BALANCE\_FREQUENCY: Ratio of last 12 months with balance
* PURCHASES: Total purchase amount spent during last 12 months
* ONEOFF\_PURCHASES: Total amount of one-off purchases
* INSTALLMENTS\_PURCHASES: Total amount of installment purchases
* CASH\_ADVANCE: Total cash-advance amount
* PURCHASES\_ FREQUENCY: Frequency of purchases (Percent of months with at least one purchase)
* ONEOFF\_PURCHASES\_FREQUENCY: Frequency of one-off-purchases
* PURCHASES\_INSTALLMENTS\_FREQUENCY: Frequency of installment purchases
* CASH\_ADVANCE\_ FREQUENCY: Cash-Advance frequency
* AVERAGE\_PURCHASE\_TRX: Average amount per purchase transaction
* CASH\_ADVANCE\_TRX: Average amount per cash-advance transaction
* PURCHASES\_TRX: Average amount per purchase transaction
* CREDIT\_LIMIT: Credit limit
* PAYMENTS: Total payments (due amount paid by the customer to decrease their statement balance) in the period
* MINIMUM\_PAYMENTS: Total minimum payments due in the period.
* PRC\_FULL\_PAYMEN: Percentage of months with full payment of the due statement balance
* TENURE: Number of months as a customer

We will look at the Python code and the process done in Credit Card Segmentation through Python Programming.

**Python Code:**

*# Importing libraries*

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**from** **math** **import** \*

**import** **sys**

*# I am going to display only 7 columns to avoid problems of overlapping when publishing in web*

pd.set\_option('display.max\_columns', 7)

*# Importing visualization libraries*

**import** **matplotlib.pyplot** **as** **plt**

**import** **matplotlib.pylab** **as** **pylab**

**import** **seaborn** **as** **sns**

**import** **missingno** **as** **msno**

plt.style.use( 'ggplot' )

%**matplotlib** inline

*#Import preprocessing libraries*

**from** **sklearn.preprocessing** **import** MinMaxScaler , StandardScaler, Imputer, LabelEncoder

*# Ignore warnings*

**import** **warnings**

warnings.filterwarnings('ignore')

*# Importing clustering libraries*

**from** **sklearn.cluster** **import** KMeans

In [2]:

*# Reading the data*

df = pd.read\_csv('C:\Users\Hector\Python\_blog\CC\_GENERAL.csv',index\_col=0)

In [3]:

*# Here are the columns.*

list(df.columns)

*# All fields are the typical that you may expect in a CC data set.*

*#---------------*

*#DATA DICTIONARY*

*#---------------*

*#Credit card holder ID*

*#Monthly average balance (based on daily balance averages)*

*#Ratio of last 12 months with balance*

*#Total purchase amount spent during last 12 months*

*#Total amount of one-off purchases*

*#Total amount of installment purchases*

*#Total cash-advance amount*

*#Frequency of purchases (percentage of months with at least one purchase)*

*#Frequency of one-off-purchases*

*#Frequency of installment purchases*

*#Cash-Advance frequency*

*#Average amount per cash-advance transaction*

*#Average amount per purchase transaction*

*#Credit limit*

*#Total payments(due amount paid by the customer to decrease their statement balance) in the period*

*#Total minimum payments due in the period.*

*#Percentage of months with full payment of the due statement balance*

*#Number of months as a customer*

Out[3]:

['BALANCE',

'BALANCE\_FREQUENCY',

'PURCHASES',

'ONEOFF\_PURCHASES',

'INSTALLMENTS\_PURCHASES',

'CASH\_ADVANCE',

'PURCHASES\_FREQUENCY',

'ONEOFF\_PURCHASES\_FREQUENCY',

'PURCHASES\_INSTALLMENTS\_FREQUENCY',

'CASH\_ADVANCE\_FREQUENCY',

'CASH\_ADVANCE\_TRX',

'PURCHASES\_TRX',

'CREDIT\_LIMIT',

'PAYMENTS',

'MINIMUM\_PAYMENTS',

'PRC\_FULL\_PAYMENT',

'TENURE']

In [4]:

df.head()

Out[4]:

|  | **BALANCE** | **BALANCE\_FREQUENCY** | **PURCHASES** | **…** | **MINIMUM\_PAYMENTS** | **PRC\_FULL\_PAYMENT** | **TENURE** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CUST\_ID** |  |  |  |  |  |  |  |
| **C10001** | 40.900749 | 0.818182 | 95.40 | … | 139.509787 | 0.000000 | 12 |
| **C10002** | 3202.467416 | 0.909091 | 0.00 | … | 1072.340217 | 0.222222 | 12 |
| **C10003** | 2495.148862 | 1.000000 | 773.17 | … | 627.284787 | 0.000000 | 12 |
| **C10004** | 1666.670542 | 0.636364 | 1499.00 | … | NaN | 0.000000 | 12 |
| **C10005** | 817.714335 | 1.000000 | 16.00 | … | 244.791237 | 0.000000 | 12 |

5 rows × 17 columns

In [5]:

*# We have 17 features in our dataframe*

df.shape

Out[5]:

(8950, 17)

In [6]:

*#we have 8950 rows and some nulls*

df.info()

<class 'pandas.core.frame.DataFrame'>

Index: 8950 entries, C10001 to C19190

Data columns (total 17 columns):

BALANCE 8950 non-null float64

BALANCE\_FREQUENCY 8950 non-null float64

PURCHASES 8950 non-null float64

ONEOFF\_PURCHASES 8950 non-null float64

INSTALLMENTS\_PURCHASES 8950 non-null float64

CASH\_ADVANCE 8950 non-null float64

PURCHASES\_FREQUENCY 8950 non-null float64

ONEOFF\_PURCHASES\_FREQUENCY 8950 non-null float64

PURCHASES\_INSTALLMENTS\_FREQUENCY 8950 non-null float64

CASH\_ADVANCE\_FREQUENCY 8950 non-null float64

CASH\_ADVANCE\_TRX 8950 non-null int64

PURCHASES\_TRX 8950 non-null int64

CREDIT\_LIMIT 8949 non-null float64

PAYMENTS 8950 non-null float64

MINIMUM\_PAYMENTS 8637 non-null float64

PRC\_FULL\_PAYMENT 8950 non-null float64

TENURE 8950 non-null int64

dtypes: float64(14), int64(3)

memory usage: 1.2+ MB

In [7]:

df.describe()

Out[7]:

|  | **BALANCE** | **BALANCE\_FREQUENCY** | **PURCHASES** | **…** | **MINIMUM\_PAYMENTS** | **PRC\_FULL\_PAYMENT** | **TENURE** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 8950.000000 | 8950.000000 | 8950.000000 | … | 8637.000000 | 8950.000000 | 8950.000000 |
| **mean** | 1564.474828 | 0.877271 | 1003.204834 | … | 864.206542 | 0.153715 | 11.517318 |
| **std** | 2081.531879 | 0.236904 | 2136.634782 | … | 2372.446607 | 0.292499 | 1.338331 |
| **min** | 0.000000 | 0.000000 | 0.000000 | … | 0.019163 | 0.000000 | 6.000000 |
| **25%** | 128.281915 | 0.888889 | 39.635000 | … | 169.123707 | 0.000000 | 12.000000 |
| **50%** | 873.385231 | 1.000000 | 361.280000 | … | 312.343947 | 0.000000 | 12.000000 |
| **75%** | 2054.140036 | 1.000000 | 1110.130000 | … | 825.485459 | 0.142857 | 12.000000 |
| **max** | 19043.138560 | 1.000000 | 49039.570000 | … | 76406.207520 | 1.000000 | 12.000000 |

8 rows × 17 columns

##### *CHECKING NULLS*

In [8]:

*# Checking nulls*

df.isnull().sum()

*# We have nulls in credit limit and minimum payments*

*# I will imput the values later.*

Out[8]:

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]:

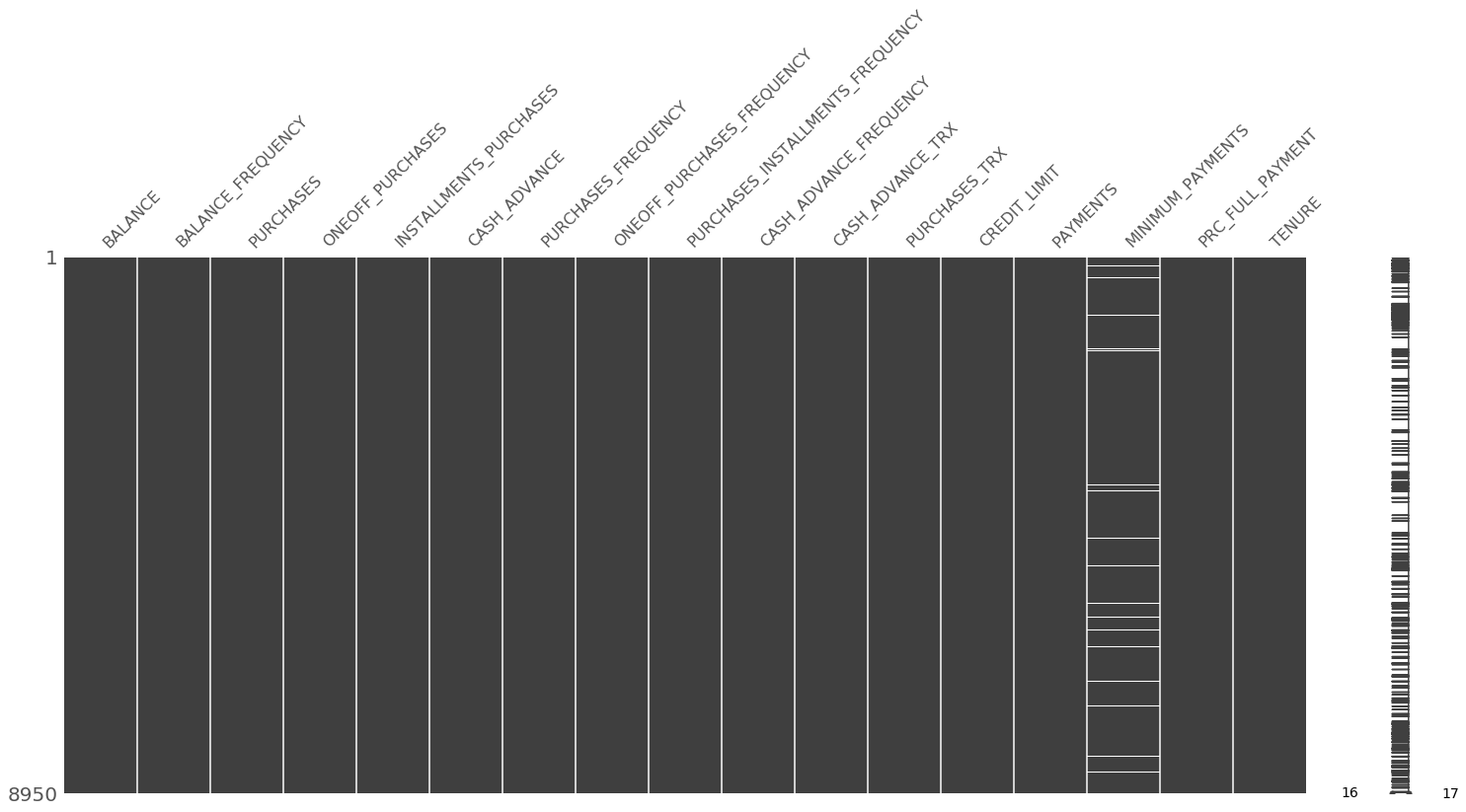
*# checking for nulls*

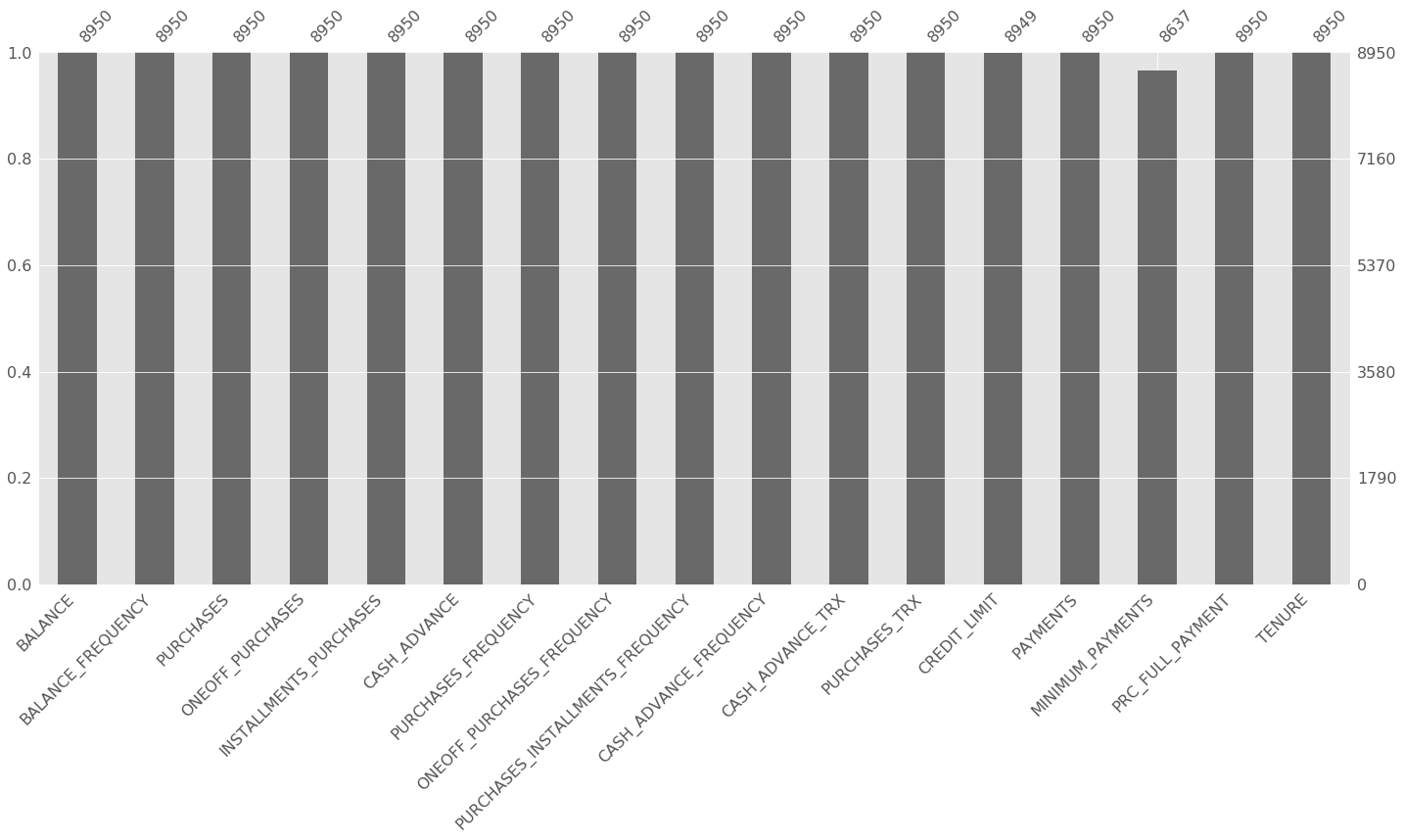
*# percentage of nan*

*# number of nan*

msno.matrix(df);

msno.bar(df);

****

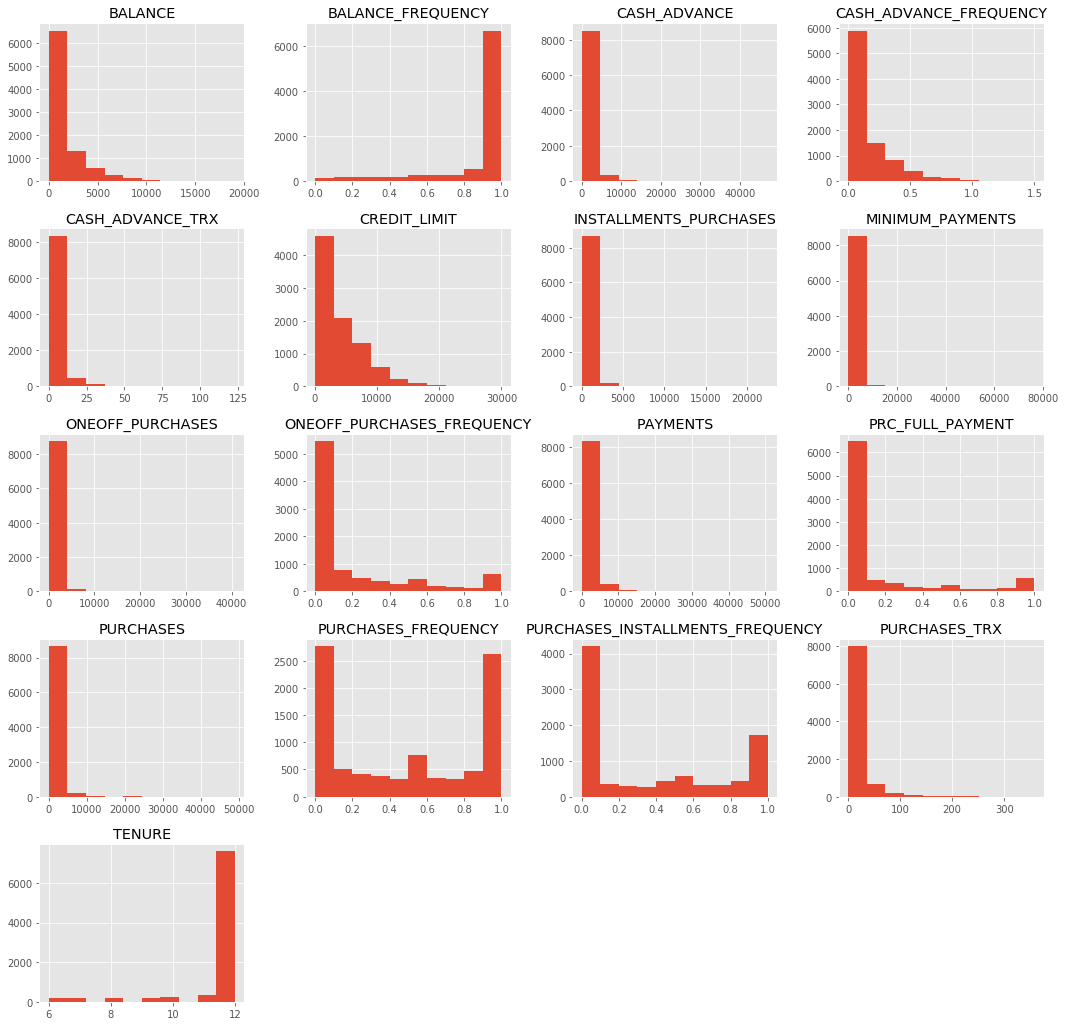
****

##### *EXPLORATORY DATA ANALYSIS*

In [10]:

*# EXPORATORY DATA ANALYSIS*

df.hist(figsize=(18,18));

****

In [11]:

*#let´s see how are distributed the frequency variables*

df[['BALANCE\_FREQUENCY',

'PURCHASES\_FREQUENCY',

'ONEOFF\_PURCHASES\_FREQUENCY',

'PURCHASES\_INSTALLMENTS\_FREQUENCY',

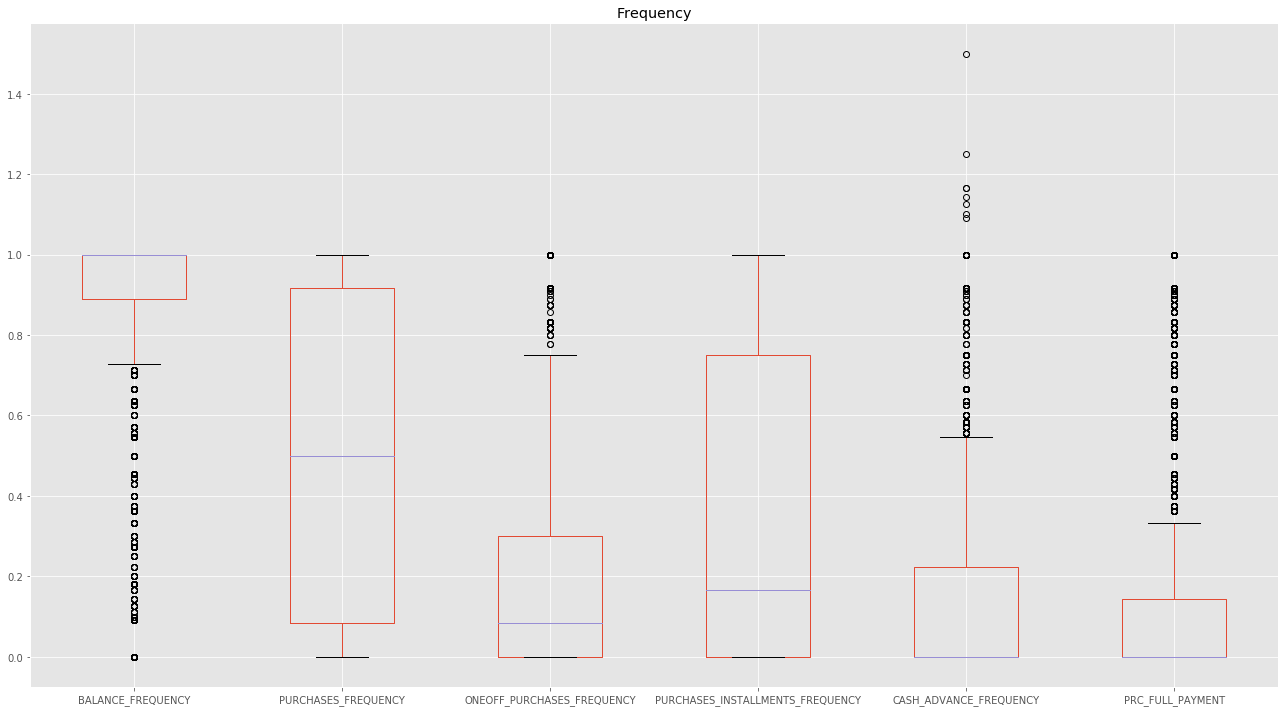
'CASH\_ADVANCE\_FREQUENCY',

'PRC\_FULL\_PAYMENT']].plot.box(figsize=(18,10),title='Frequency',legend=True);

plt.tight\_layout()

*# We have data on Cash\_advance\_frequency that is wrong. I will clean the dataset later.*

*# There are also many outliers, but we will keep then for now*

****

In [12]:

*#let´s see how are distributed the numeric variables*

df[['BALANCE',

'PURCHASES',

'ONEOFF\_PURCHASES',

'INSTALLMENTS\_PURCHASES',

'CASH\_ADVANCE',

'CREDIT\_LIMIT',

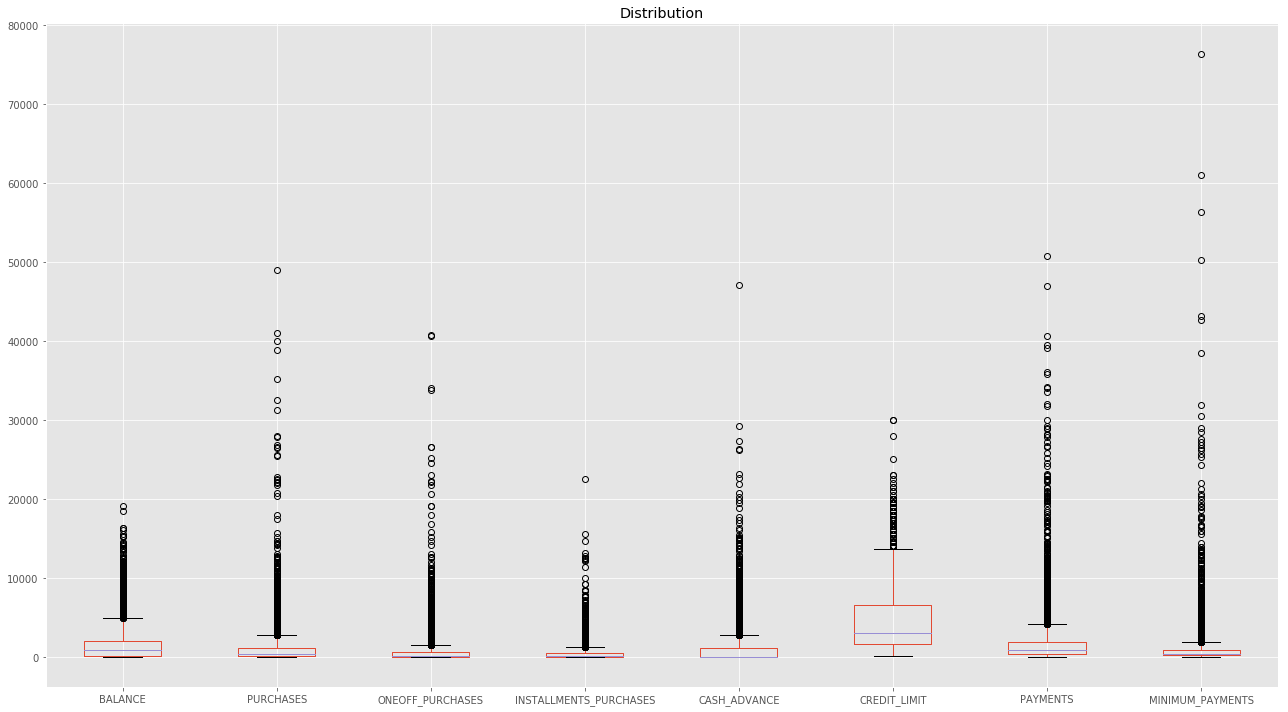
'PAYMENTS',

'MINIMUM\_PAYMENTS'

]].plot.box(figsize=(18,10),title='Distribution',legend=True);

plt.tight\_layout()

*# There are also many outliers, but we will keep them for now*

****

In [13]:

*#let´s see how are distributed the numeric variables*

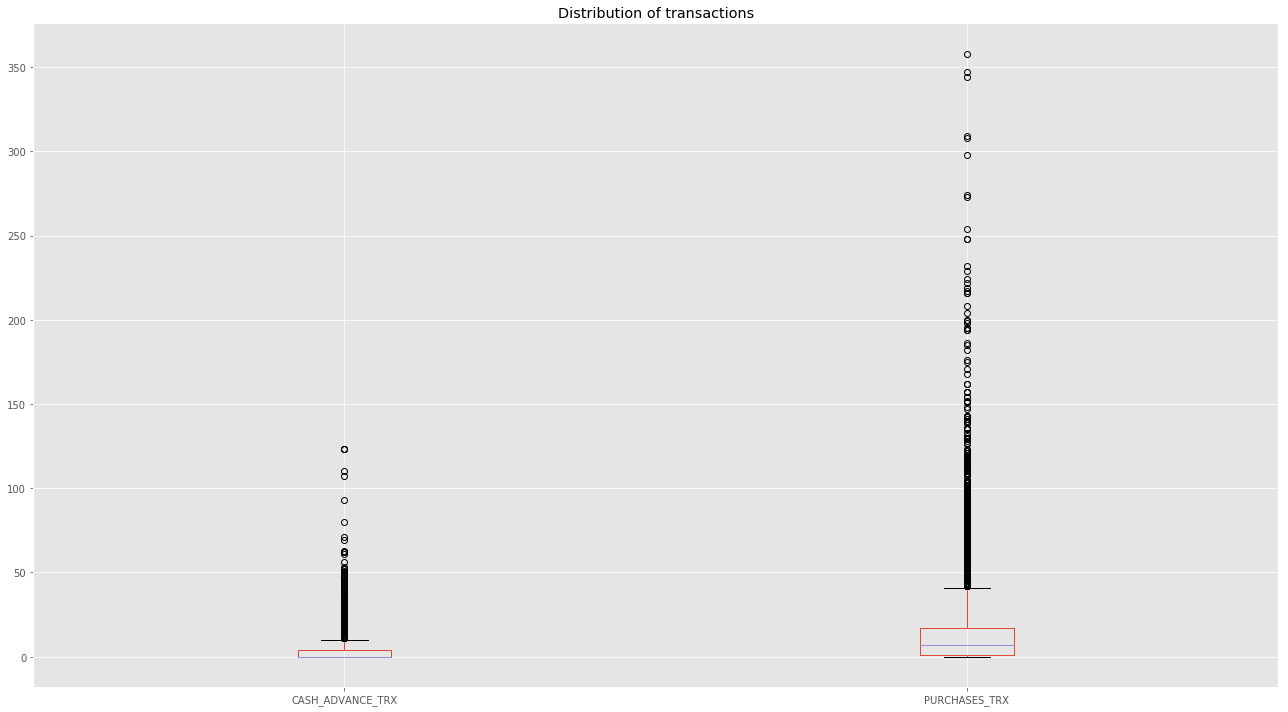
df[[ 'CASH\_ADVANCE\_TRX',

'PURCHASES\_TRX'

]].plot.box(figsize=(18,10),title='Distribution of transactions',legend=True);

plt.tight\_layout()

*# There are also many outliers, but we will keep them for now*

****

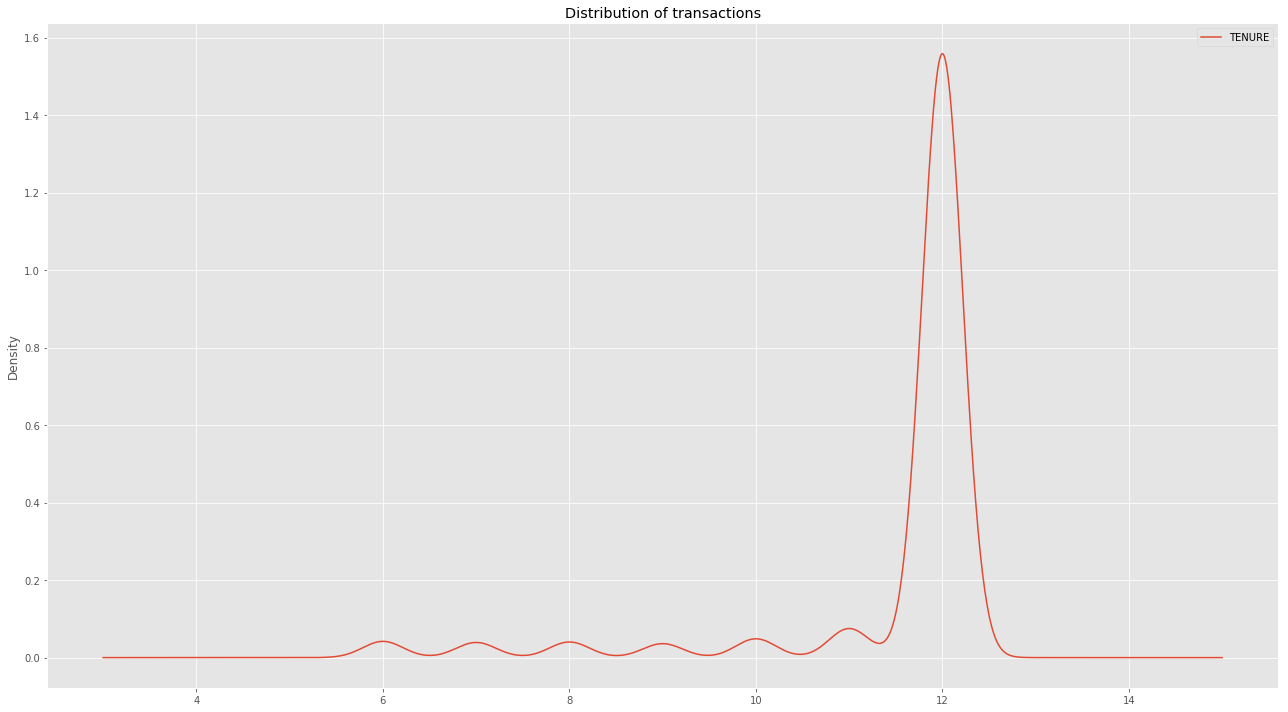
In [14]:

*#let´s see how is distributed the tenure*

df[['TENURE'

]].plot.kde(figsize=(18,10),title='Distribution of transactions',legend=True);

plt.tight\_layout()

****

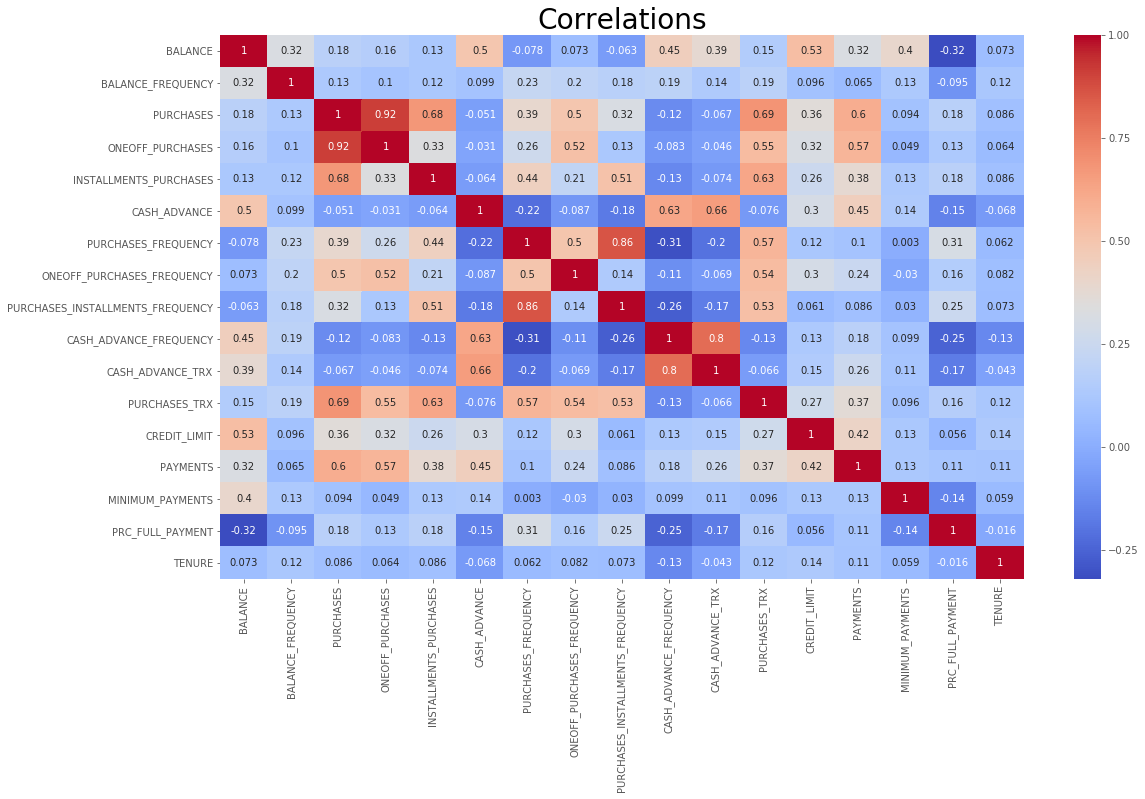
In [15]:

*#Lets take a look at how the variables are correlated*

plt.figure(figsize=(18,10))

sns.heatmap(df.corr(),cmap='coolwarm',annot=True);

plt.title('Correlations', size = 28);

****

##### *DATA PREPARATION*

##### *CLEANING*

In [16]:

*# Lets clean the data (inputing values and eliminating wrong data) before the segmentation*

df.loc[(df['CASH\_ADVANCE\_FREQUENCY']>1)]

*# we have 8 records for which the frequency is higher that 1. I will eliminate these records*

Out[16]:

|  | **BALANCE** | **BALANCE\_FREQUENCY** | **PURCHASES** | **…** | **MINIMUM\_PAYMENTS** | **PRC\_FULL\_PAYMENT** | **TENURE** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CUST\_ID** |  |  |  |  |  |  |  |
| **C10708** | 5656.069801 | 1.000000 | 362.36 | … | 2036.877611 | 0.0 | 8 |
| **C11680** | 2876.009336 | 1.000000 | 152.61 | … | 584.926336 | 0.0 | 6 |
| **C12629** | 5906.184924 | 1.000000 | 141.80 | … | 919.289675 | 0.0 | 8 |
| **C12684** | 7801.511533 | 1.000000 | 231.40 | … | 1522.496755 | 0.0 | 10 |
| **C13127** | 3846.742530 | 1.000000 | 0.00 | … | 538.346874 | 0.0 | 6 |
| **C13347** | 5709.486507 | 0.833333 | 0.00 | … | 1299.463370 | 0.0 | 6 |
| **C18273** | 1917.895730 | 1.000000 | 285.07 | … | 556.449635 | 0.0 | 11 |
| **C18588** | 3857.562230 | 1.000000 | 0.00 | … | 538.396872 | 0.0 | 7 |

8 rows × 17 columns

In [17]:

*# dropping the records with frequency higher that 1*

df = df[(df[['CASH\_ADVANCE\_FREQUENCY']] <= 1).all(axis=1)]

In [18]:

df.shape

Out[18]:

(8942, 17)

##### *INPUTTING NULL VALUES AND STANDARDIZATION*

##### *INPUTING NULL VALUES*

In [19]:

*# Imputing values in 'MINIMUM\_PAYMENTS' and 'CREDIT\_LIMIT'*

*#I will use the median to input the values*

df['MINIMUM\_PAYMENTS'].fillna(df['MINIMUM\_PAYMENTS'].median(),inplace=True)

df['CREDIT\_LIMIT'].fillna(df['CREDIT\_LIMIT'].median(),inplace=True)

*# I get rid of Customer Id as wee don´t need it*

*#df.reset\_index(inplace=True)*

*#df.drop('CUST\_ID',inplace=True,axis=1)*

df.head()

Out[19]:

|  | **BALANCE** | **BALANCE\_FREQUENCY** | **PURCHASES** | **…** | **MINIMUM\_PAYMENTS** | **PRC\_FULL\_PAYMENT** | **TENURE** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CUST\_ID** |  |  |  |  |  |  |  |
| **C10001** | 40.900749 | 0.818182 | 95.40 | … | 139.509787 | 0.000000 | 12 |
| **C10002** | 3202.467416 | 0.909091 | 0.00 | … | 1072.340217 | 0.222222 | 12 |
| **C10003** | 2495.148862 | 1.000000 | 773.17 | … | 627.284787 | 0.000000 | 12 |
| **C10004** | 1666.670542 | 0.636364 | 1499.00 | … | 311.637186 | 0.000000 | 12 |
| **C10005** | 817.714335 | 1.000000 | 16.00 | … | 244.791237 | 0.000000 | 12 |

5 rows × 17 columns

##### *STANDARDIZATION*

In [20]:

*# Before using K-Means, as in K-means we optimize the sum of squared distances between the observations and their centroids*

*# and as some varibles are expresed in different variables i.e frequencies, currency amount and number of transactions,*

*# we need to standardize.*

*# I am going to leave this section commented, because later on I would like to explore the centroids, but standardizing*

*# the data will give us better results.*

*# Then, we would follow the analysis with dataframe df\_scaled instead of df.*

*#Standardization*

*# Create the scaler object with a range of 0-1*

scaler = MinMaxScaler(feature\_range=(0, 1))

df\_columns=df.columns

*# Fit on the data and transform*

df\_s=scaler.fit\_transform(df.values)

*# Create the new dataframe*

df\_scaled = pd.DataFrame(df\_s,columns=df\_columns)

*#df.head()*

df\_scaled.head()

Out[20]:

|  | **BALANCE** | **BALANCE\_FREQUENCY** | **PURCHASES** | **…** | **MINIMUM\_PAYMENTS** | **PRC\_FULL\_PAYMENT** | **TENURE** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.002148 | 0.818182 | 0.001945 | … | 0.001826 | 0.000000 | 1.0 |
| **1** | 0.168169 | 0.909091 | 0.000000 | … | 0.014034 | 0.222222 | 1.0 |
| **2** | 0.131026 | 1.000000 | 0.015766 | … | 0.008210 | 0.000000 | 1.0 |
| **3** | 0.087521 | 0.636364 | 0.030567 | … | 0.004078 | 0.000000 | 1.0 |
| **4** | 0.042940 | 1.000000 | 0.000326 | … | 0.003204 | 0.000000 | 1.0 |

5 rows × 17 columns

##### *K-MEANS CLUSTERING*

##### *CALCULATING THE OPTIMAL NUMBER OF CLUSTERS*

In [21]:

*# K MEANS CLUSTERING*

*# Before using K means, i am going to determine the optimal number of clusters*

*# i will use the Elbow method to find a good number of clusters*

inertia = []

**for** k **in** range(1, 20):

kmeans = KMeans(n\_clusters=k, random\_state=1).fit(df)

inertia.append(np.sqrt(kmeans.inertia\_))

In [22]:

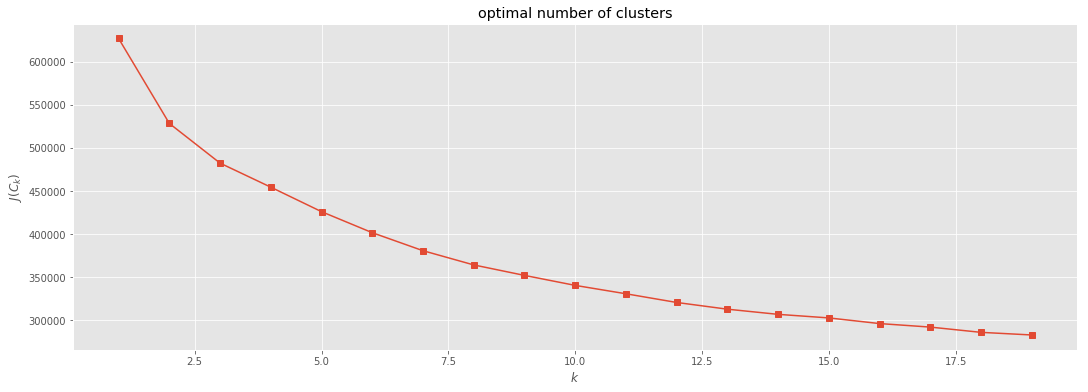
plt.figure(figsize=(18,6))

plt.title('optimal number of clusters')

plt.plot(range(1, 20), inertia, marker='s');

plt.xlabel('$k$')

plt.ylabel('$J(C\_k)$');

****

##### *CALCULATING K-MEANS AND THE CENTROIDS*

In [23]:

*# It seems that the optimal number of clusters is between 7.5 and 8.*

*# I am going to take 8 for the analysis*

kmeans = KMeans(n\_clusters=8)

In [24]:

*# applying kmeans*

kmeans.fit(df)

Out[24]:

KMeans(algorithm='auto', copy\_x=True, init='k-means++', max\_iter=300,

n\_clusters=8, n\_init=10, n\_jobs=1, precompute\_distances='auto',

random\_state=None, tol=0.0001, verbose=0)

In [25]:

*# Calculating the centroids*

centroids=kmeans.cluster\_centers\_

centroids

Out[25]:

array([[7.80976721e+02, 8.50103225e-01, 4.99055779e+02, 2.41365391e+02,

2.57992818e+02, 4.42438153e+02, 4.51465876e-01, 1.29670669e-01,

3.47330644e-01, 1.07517503e-01, 2.19307972e+00, 9.37201300e+00,

2.12232598e+03, 8.60285421e+02, 5.01103448e+02, 1.44047149e-01,

1.13666603e+01],

[5.18347658e+03, 8.76681796e-01, 1.45105673e+03, 8.68528319e+02,

5.82723097e+02, 1.08991158e+04, 3.51573558e-01, 1.93545743e-01,

2.73857425e-01, 4.97731097e-01, 2.07522124e+01, 1.83008850e+01,

1.08491150e+04, 1.55667252e+04, 2.07362398e+03, 1.45884186e-01,

1.16814159e+01],

[2.36333884e+03, 9.70790522e-01, 6.55829643e+03, 4.42557614e+03,

2.13449021e+03, 5.39565300e+02, 9.12955519e-01, 7.02838118e-01,

7.05029667e-01, 7.16009410e-02, 1.94395280e+00, 7.36194690e+01,

8.02463127e+03, 6.50836341e+03, 1.14587468e+03, 3.20409711e-01,

1.19262537e+01],

[5.60569286e+03, 9.43575135e-01, 1.60729190e+03, 9.77871195e+02,

6.29568827e+02, 2.09559895e+03, 5.62101097e-01, 3.30390166e-01,

4.06539969e-01, 2.11286520e-01, 5.34734513e+00, 2.32013274e+01,

1.35319690e+04, 2.58764813e+03, 1.77752020e+03, 1.17469575e-01,

1.18761062e+01],

[5.44820172e+03, 9.56126478e-01, 2.79165557e+04, 2.23543143e+04,

5.56224130e+03, 9.70110470e+02, 9.05072435e-01, 8.50000043e-01,

7.08695609e-01, 3.98550435e-02, 1.78260870e+00, 1.28217391e+02,

1.60434783e+04, 2.77616004e+04, 3.34571825e+03, 5.15316174e-01,

1.19130435e+01],

[4.01407117e+03, 9.57742708e-01, 4.98504608e+02, 2.78905471e+02,

2.19633176e+02, 3.52856813e+03, 3.11706503e-01, 1.33285347e-01,

2.13607576e-01, 3.63914596e-01, 1.00509804e+01, 8.58921569e+00,

6.34024955e+03, 2.66908847e+03, 1.54607643e+03, 2.83226324e-02,

1.14882353e+01],

[8.86588456e+02, 8.71233816e-01, 1.19676636e+03, 6.98024845e+02,

4.98964424e+02, 2.24145846e+02, 6.19020933e-01, 3.30856169e-01,

4.28702744e-01, 4.90230415e-02, 9.21591574e-01, 1.89321240e+01,

6.98267195e+03, 1.41459110e+03, 3.44918361e+02, 2.35933719e-01,

1.18086600e+01],

[4.06673126e+03, 9.88429764e-01, 1.04349473e+03, 1.19814727e+02,

9.23680000e+02, 9.20037169e+02, 4.70798873e-01, 3.84297091e-02,

4.41597818e-01, 1.03030255e-01, 3.01818182e+00, 1.88363636e+01,

4.26636364e+03, 1.58459625e+03, 2.29600155e+04, 1.51514545e-03,

1.19090909e+01]])

In [26]:

*# Creating a dataframe for the centroids*

centroids=kmeans.cluster\_centers\_

index=['C0','C1','C2','C3','C4','C5','C6','C7']

columns=df.columns

centroid\_df=pd.DataFrame(centroids,index,columns)

centroid\_df

Out[26]:

|  | **BALANCE** | **BALANCE\_FREQUENCY** | **PURCHASES** | **…** | **MINIMUM\_PAYMENTS** | **PRC\_FULL\_PAYMENT** | **TENURE** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **C0** | 780.976721 | 0.850103 | 499.055779 | … | 501.103448 | 0.144047 | 11.366660 |
| **C1** | 5183.476576 | 0.876682 | 1451.056726 | … | 2073.623975 | 0.145884 | 11.681416 |
| **C2** | 2363.338837 | 0.970791 | 6558.296431 | … | 1145.874684 | 0.320410 | 11.926254 |
| **C3** | 5605.692859 | 0.943575 | 1607.291903 | … | 1777.520195 | 0.117470 | 11.876106 |
| **C4** | 5448.201718 | 0.956126 | 27916.555652 | … | 3345.718254 | 0.515316 | 11.913043 |
| **C5** | 4014.071172 | 0.957743 | 498.504608 | … | 1546.076429 | 0.028323 | 11.488235 |
| **C6** | 886.588456 | 0.871234 | 1196.766360 | … | 344.918361 | 0.235934 | 11.808660 |
| **C7** | 4066.731264 | 0.988430 | 1043.494727 | … | 22960.015462 | 0.001515 | 11.909091 |

8 rows × 17 columns

##### *ADDING THE LABELS TO THE DATASET*

In [27]:

*#Adding the clusters to the dataframe*

df['cluster']=list(kmeans.labels\_)

In [28]:

df.head()

Out[28]:

|  | **BALANCE** | **BALANCE\_FREQUENCY** | **PURCHASES** | **…** | **PRC\_FULL\_PAYMENT** | **TENURE** | **cluster** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CUST\_ID** |  |  |  |  |  |  |  |
| **C10001** | 40.900749 | 0.818182 | 95.40 | … | 0.000000 | 12 | 0 |
| **C10002** | 3202.467416 | 0.909091 | 0.00 | … | 0.222222 | 12 | 5 |
| **C10003** | 2495.148862 | 1.000000 | 773.17 | … | 0.000000 | 12 | 6 |
| **C10004** | 1666.670542 | 0.636364 | 1499.00 | … | 0.000000 | 12 | 6 |
| **C10005** | 817.714335 | 1.000000 | 16.00 | … | 0.000000 | 12 | 0 |

5 rows × 18 columns

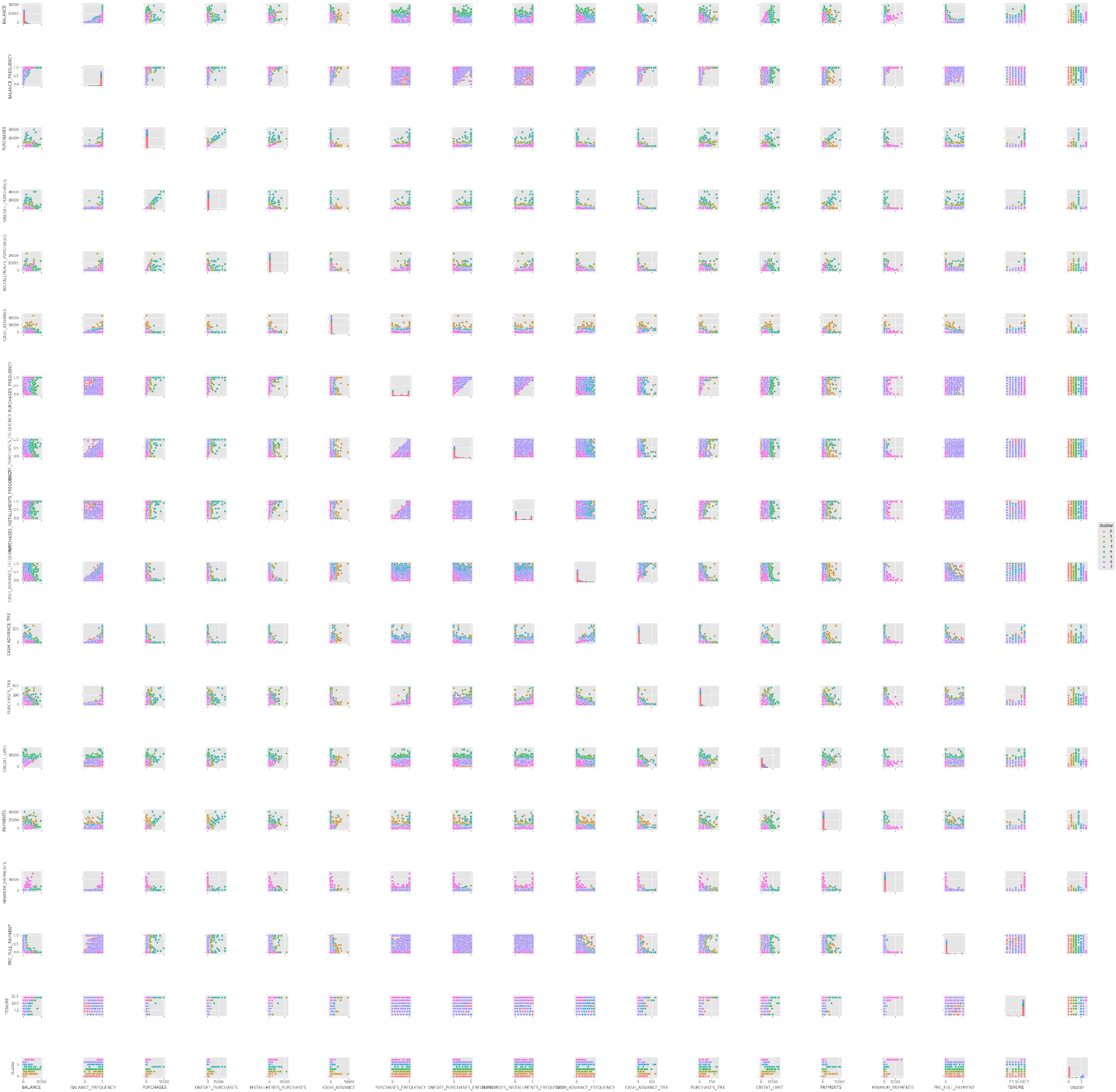
##### *VISUALIZING THE CUSTOMER SEGMENTATION*

In [29]:

sns.pairplot( df, hue="cluster")

Out[29]:

<seaborn.axisgrid.PairGrid at 0xcdef7f0>

****

##### *REFINING THE APPROACH THROUGH ITERATION*

In [30]:

*# I WILL REPEAT THE PROCESS BUT USING THE VARIABLES THAT HAVE MANAGED TO SEPARATE THE CLUSTERS MORE CLEARLY*

*# VISUALIZING,DESCRIBING AND EXPLAINING THE CLUSTERS*

best\_cols = ["BALANCE", "PURCHASES\_FREQUENCY", "CASH\_ADVANCE","INSTALLMENTS\_PURCHASES",

"CREDIT\_LIMIT", "PAYMENTS","PRC\_FULL\_PAYMENT" ]

kmeans = KMeans(n\_clusters=8, init="k-means++", n\_init=10, max\_iter=300)

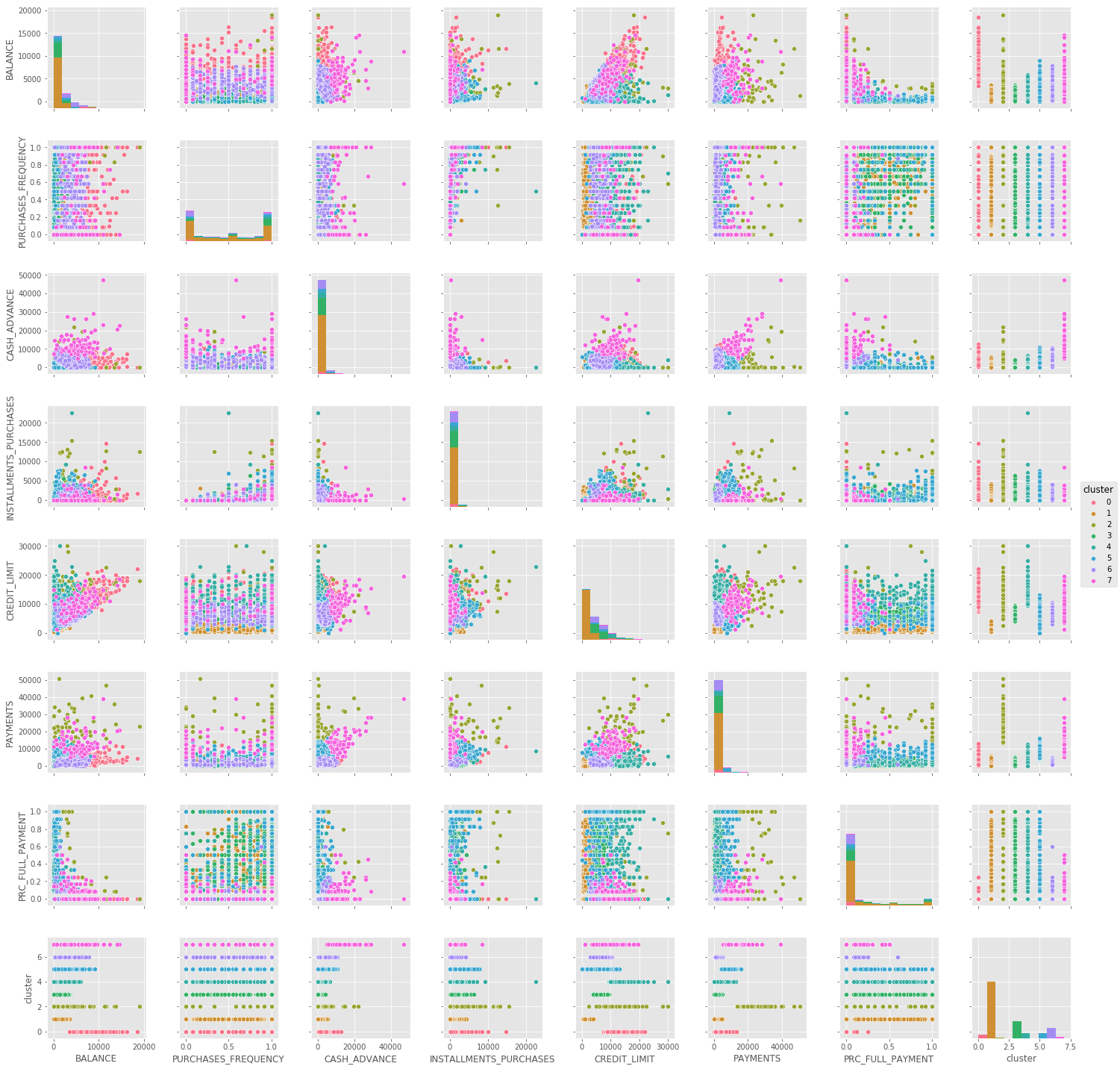
best\_vals = df[best\_cols].iloc[ :, :].values

y\_pred = kmeans.fit\_predict( best\_vals )

df["cluster"] = y\_pred

best\_cols.append("cluster")

sns.pairplot( df[ best\_cols ], hue="cluster");

****

##### *CLUSTERS EXPLANATION AND MARKETING STRATEGY*

In [31]:

*# Number of clients by cluster*

df['cluster'].value\_counts().plot.bar(figsize=(10,5), title='Customers by cluster');

df['cluster'].value\_counts()

Out[31]:

1 5105

3 1534

6 934

4 457

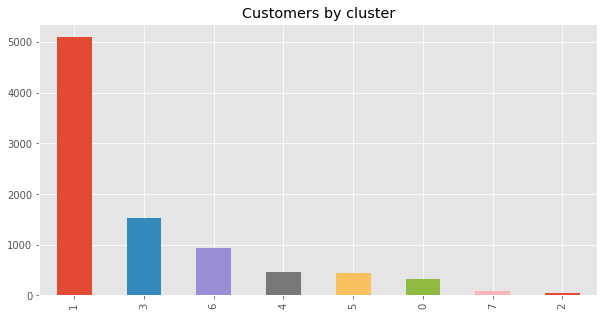
5 443

0 327

7 89

2 53

Name: cluster, dtype: int64

****

In [32]:

*# Creating a dataframe for the centroids*

centroids=kmeans.cluster\_centers\_

index=['C0','C1','C2','C3','C4','C5','C6','C7']

columns=["BALANCE", "PURCHASES\_FREQUENCY", "CASH\_ADVANCE","INSTALLMENTS\_PURCHASES",

"CREDIT\_LIMIT", "PAYMENTS","PRC\_FULL\_PAYMENT" ]

centroid\_df=pd.DataFrame(centroids,index,columns)

centroid\_df

Out[32]:

|  | **BALANCE** | **PURCHASES\_FREQUENCY** | **CASH\_ADVANCE** | **INSTALLMENTS\_PURCHASES** | **CREDIT\_LIMIT** | **PAYMENTS** | **PRC\_FULL\_PAYMENT** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **C0** | 8205.380879 | 0.460262 | 3924.869759 | 723.809725 | 12183.180428 | 3028.359434 | 0.001934 |
| **C1** | 773.115129 | 0.455137 | 427.906887 | 261.756995 | 2057.047523 | 847.174098 | 0.142505 |
| **C2** | 4352.085166 | 0.677216 | 2634.179412 | 4171.996981 | 14237.735849 | 24309.184989 | 0.430023 |
| **C3** | 749.857903 | 0.630032 | 169.718871 | 539.594948 | 6333.726443 | 1397.216418 | 0.246912 |
| **C4** | 1492.446217 | 0.676236 | 389.839950 | 742.808687 | 12519.474836 | 2213.804011 | 0.289417 |
| **C5** | 2240.620606 | 0.561823 | 2427.731424 | 972.593860 | 6302.079486 | 7069.741224 | 0.225833 |
| **C6** | 4031.995186 | 0.345592 | 2633.900572 | 258.583754 | 6174.385027 | 1546.941163 | 0.005996 |
| **C7** | 5728.519044 | 0.342924 | 12892.903721 | 510.213371 | 10436.516854 | 13088.319888 | 0.093236 |

In [33]:

*#CLUSTERS DESCRIPTION*

*#VH-very high, H- high, M- medium, L- low, VL- Very low*

*#C1 5105*

*#C3 1534*

*#C6 934*

*#C4 457*

*#C5 443*

*#C0 327*

*#C7 89*

*#C2 53*

##### *CLUSTER C0*

In [34]:

*#CLUSTER C0 -->327, VIP clients, strategy find ways for them to buy more. Mileage program*

*#----------*

*#BALANCE--> VH*

*#PURCHASE FREQUENCY-->M*

*#CASH ADVANCE-->H*

*#INSTALLMENTS PURCHASES-->H*

*#CREDIT LIMIT-->VH*

*#PAYMENTS-->M*

*#FULL PAYMENT-->L*

##### *CLUSTER C0 PERSONA*

In [35]:

*#let´s characterize the Persona in the Cluster CO*

cluster\_C0=df[df['cluster']==0]

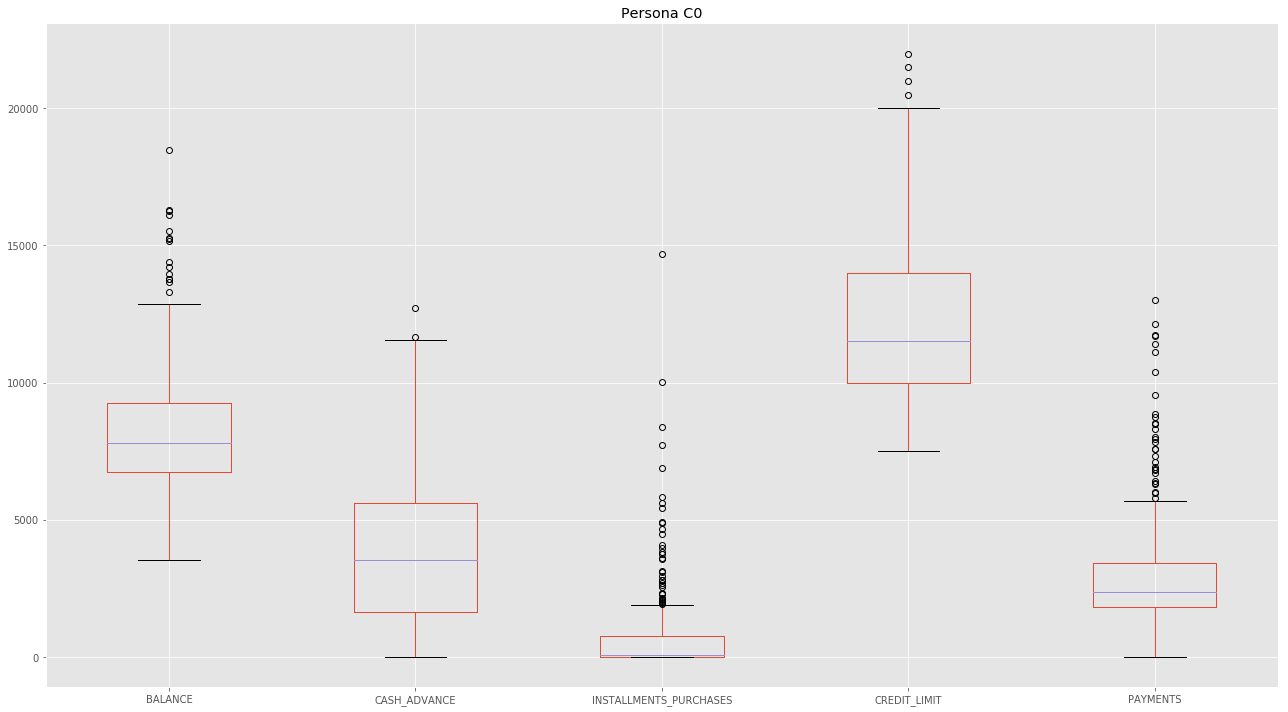
cluster\_C0[['BALANCE','CASH\_ADVANCE','INSTALLMENTS\_PURCHASES',

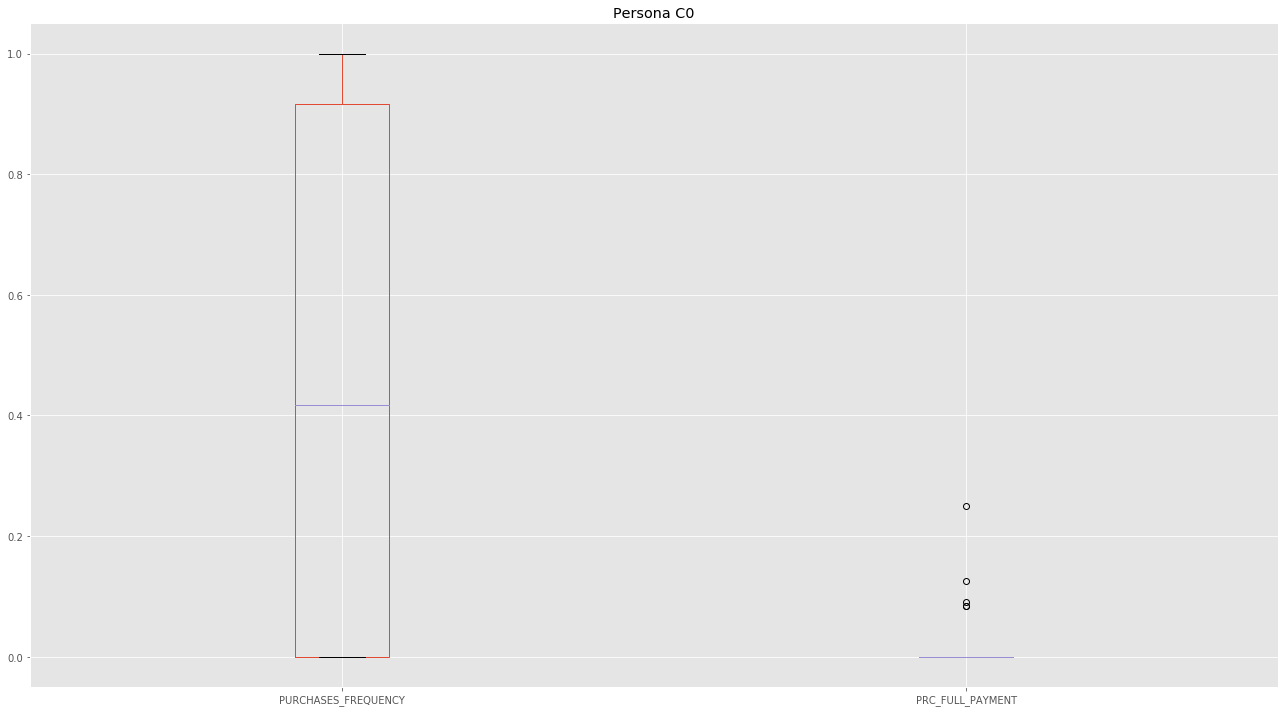
'CREDIT\_LIMIT','PAYMENTS']].plot.box(figsize=(18,10),by='cluster',title='Persona C0',legend=True);

plt.tight\_layout()

cluster\_C0[['PURCHASES\_FREQUENCY','PRC\_FULL\_PAYMENT']].plot.box(figsize=(18,10),title='Persona C0',legend=True);

plt.tight\_layout()

****

****

##### *CLUSTER C1*

In [36]:

*#CLUSTER C1-->5105, bulk of the customers. Try for them to use the card more frequently. Rewards program.*

*#----------*

*#BALANCE--> L*

*#PURCHASE FREQUENCY-->M*

*#CASH ADVANCE-->L*

*#INSTALLMENTS PURCHASES-->L*

*#CREDIT LIMIT-->L*

*#PAYMENTS-->L*

*#FULL PAYMENT-->M*

##### *CLUSTER C1 PERSONA*

In [37]:

*#let´s characterize the Persona in the Cluster C1*

cluster\_C1=df[df['cluster']==1]

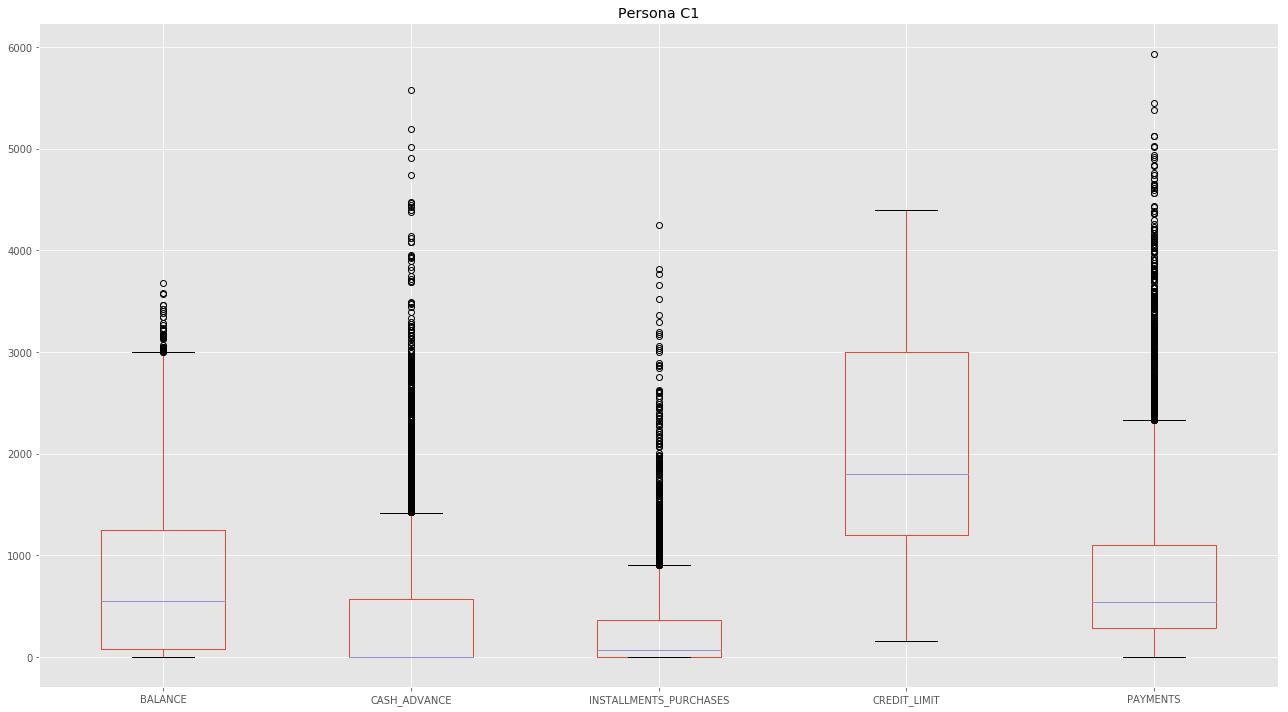
cluster\_C1[['BALANCE','CASH\_ADVANCE','INSTALLMENTS\_PURCHASES',

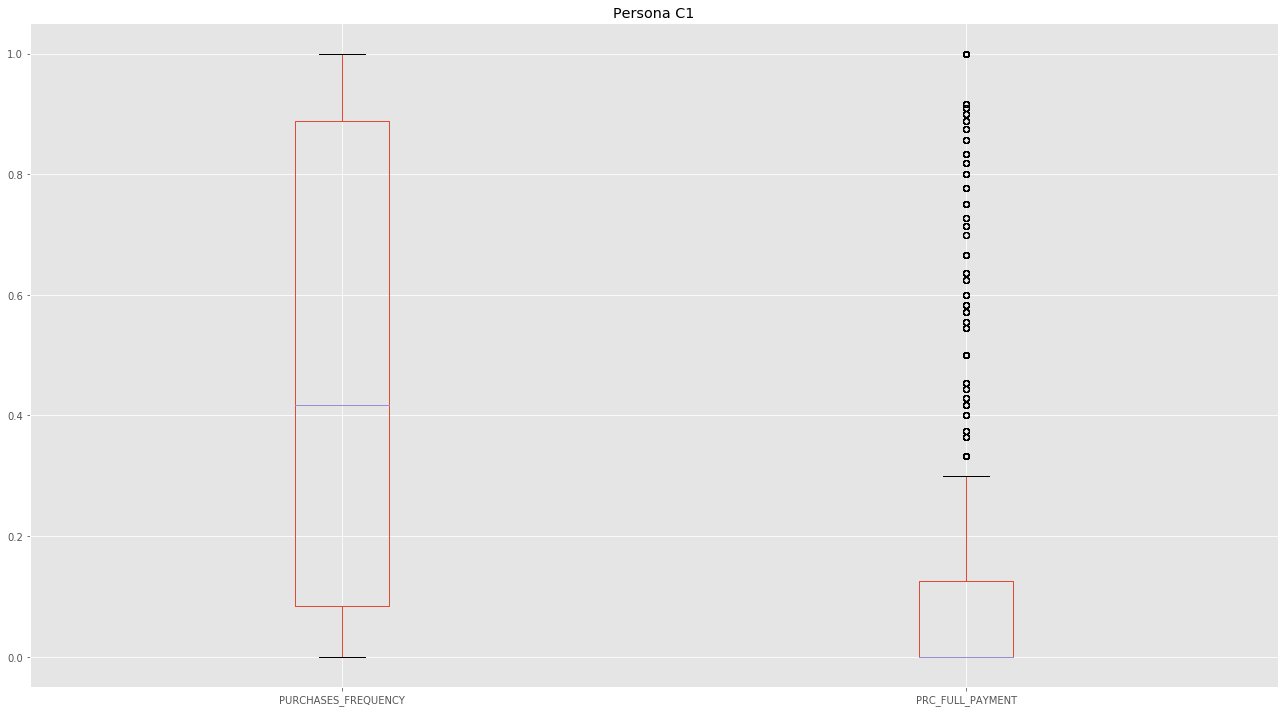
'CREDIT\_LIMIT','PAYMENTS']].plot.box(figsize=(18,10),by='cluster',title='Persona C1',legend=True);

plt.tight\_layout()

cluster\_C1[['PURCHASES\_FREQUENCY','PRC\_FULL\_PAYMENT']].plot.box(figsize=(18,10),title='Persona C1',legend=True);

plt.tight\_layout()

****

****

##### *CLUSTER C2*

In [38]:

*#CLUSTER C2-->53 VIP CUSTOMERS THAT BUY FREQUENTLY AND THE CARD TO WITHDROW MONEY. INCREASE THE CREDIT LIMIT AND OFFER T*

*#THEM LOANS*

*#----------*

*#BALANCE--> H*

*#PURCHASE FREQUENCY-->H*

*#CASH ADVANCE-->M*

*#INSTALLMENTS PURCHASES-->VH*

*#CREDIT LIMIT-->VH*

*#PAYMENTS-->VH*

*#FULL PAYMENT-->VH*

##### *CLUSTER C2 PERSONA*

In [39]:

*#let´s characterize the Persona in the Cluster C2*

cluster\_C2=df[df['cluster']==2]

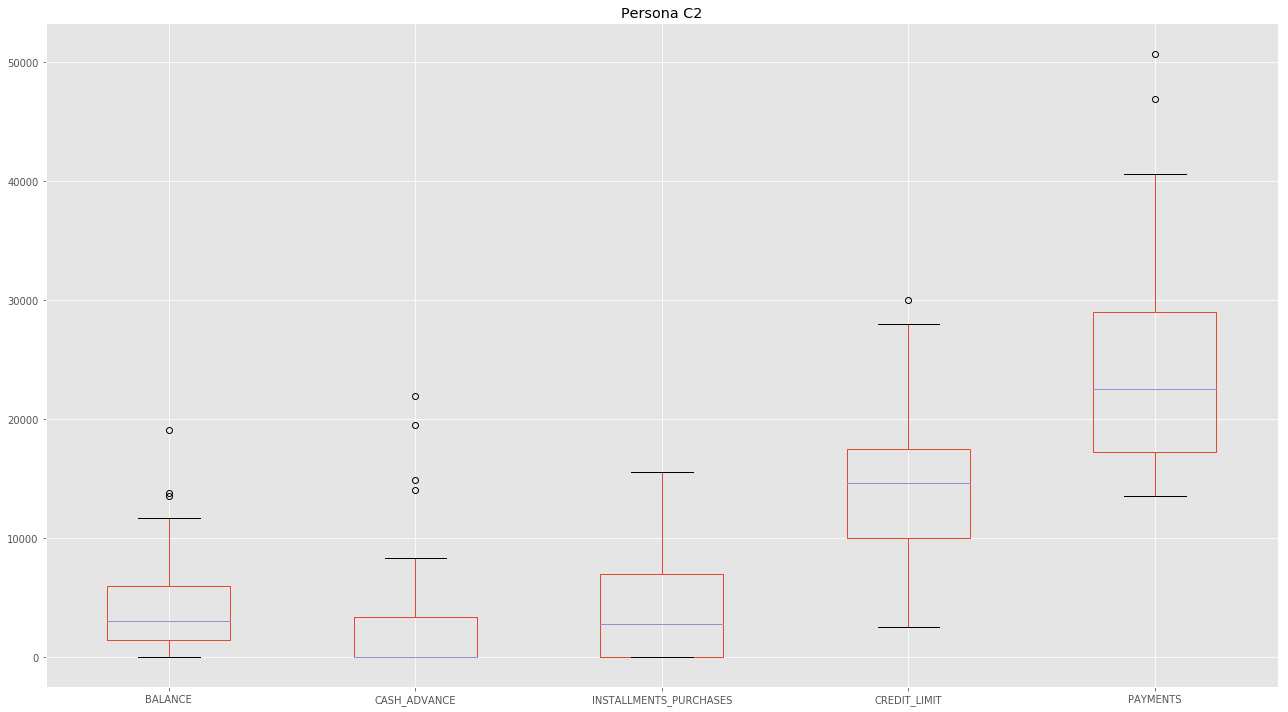
cluster\_C2[['BALANCE','CASH\_ADVANCE','INSTALLMENTS\_PURCHASES',

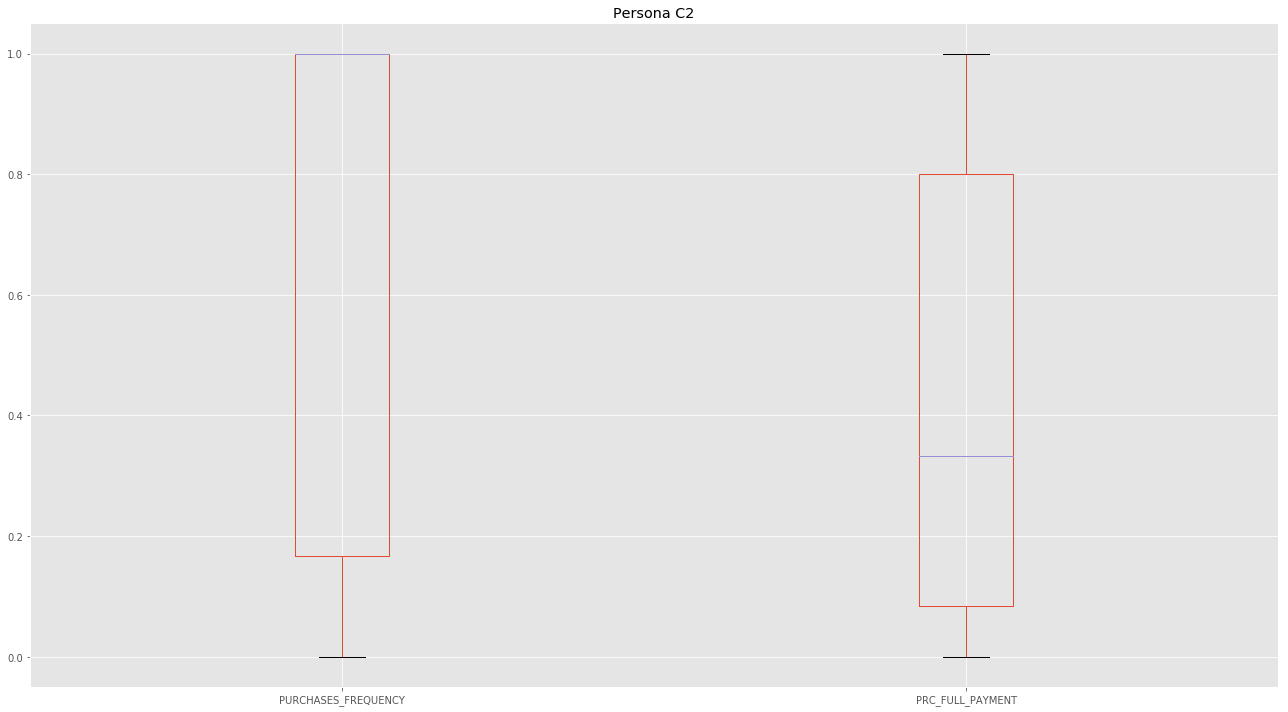
'CREDIT\_LIMIT','PAYMENTS']].plot.box(figsize=(18,10),by='cluster',title='Persona C2',legend=True);

plt.tight\_layout()

cluster\_C2[['PURCHASES\_FREQUENCY','PRC\_FULL\_PAYMENT']].plot.box(figsize=(18,10),title='Persona C2',legend=True);

plt.tight\_layout()

****

****

##### *CLUSTER C3*

In [40]:

*#CLUSTER C3 -->1534, THESE ARE TRANSACTORS, NOT VERY PROFITABLE*

*#----------*

*#BALANCE--> L*

*#PURCHASE FREQUENCY-->H*

*#CASH ADVANCE-->L*

*#INSTALLMENTS PURCHASES-->M*

*#CREDIT LIMIT-->H*

*#PAYMENTS-->L*

*#FULL PAYMENT-->H*

##### *CLUSTER C3 PERSONA*

In [41]:

*#let´s characterize the Persona in the Cluster C3*

cluster\_C3=df[df['cluster']==3]

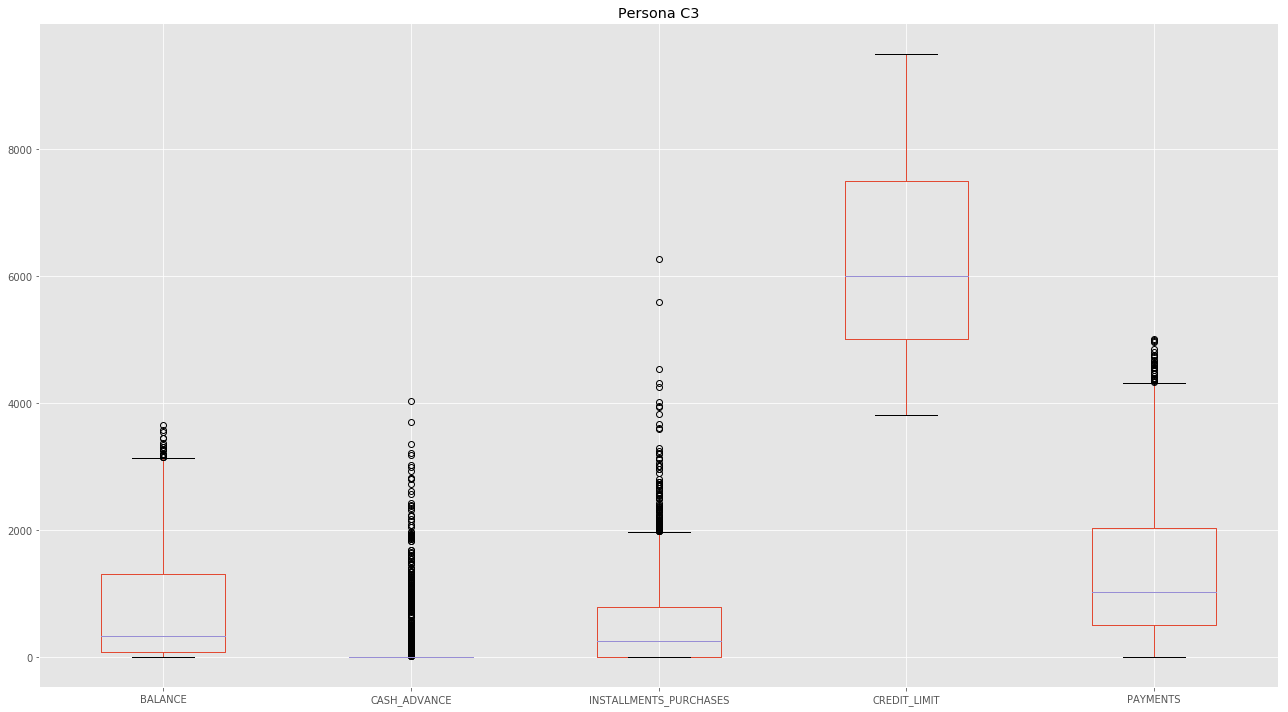
cluster\_C3[['BALANCE','CASH\_ADVANCE','INSTALLMENTS\_PURCHASES',

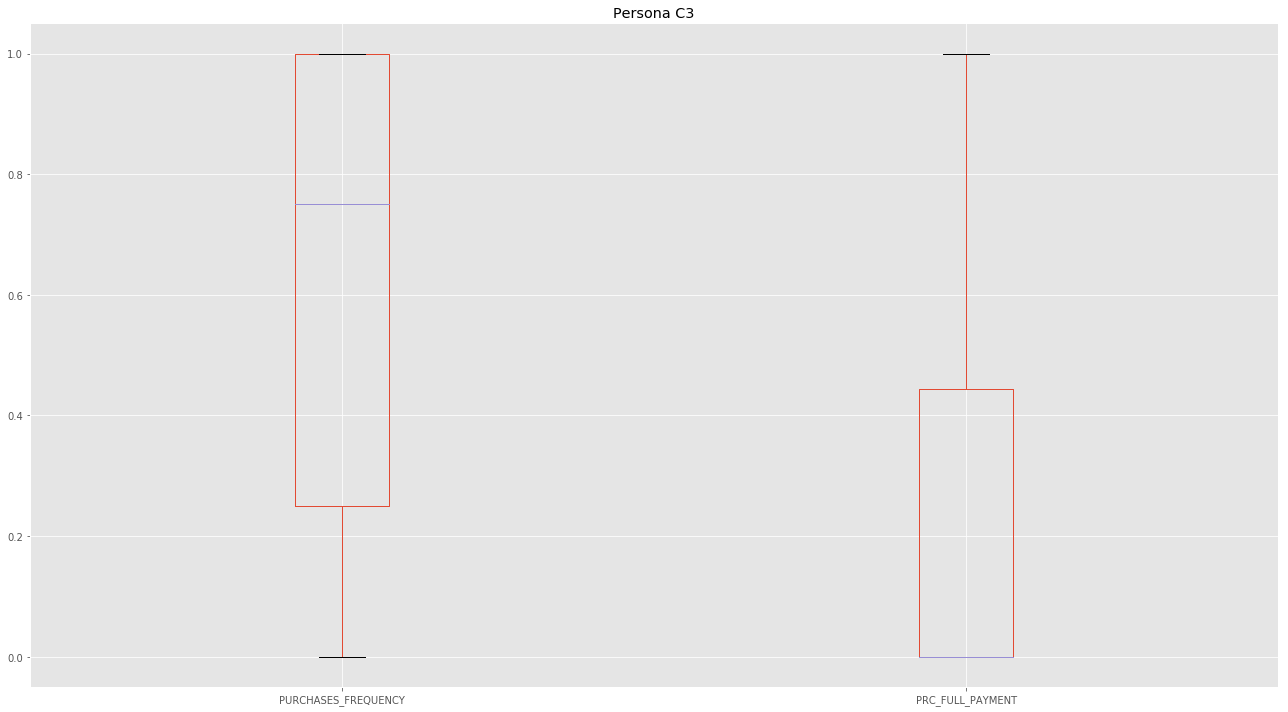
'CREDIT\_LIMIT','PAYMENTS']].plot.box(figsize=(18,10),by='cluster',title='Persona C3',legend=True);

plt.tight\_layout()

cluster\_C3[['PURCHASES\_FREQUENCY','PRC\_FULL\_PAYMENT']].plot.box(figsize=(18,10),title='Persona C3',legend=True);

plt.tight\_layout()

****

****

##### *CUSTER C4*

In [42]:

*#CLUSTER C4 -->934, THESE CUSTOMERS USE THE CREDIT CARD FOR INSTALLMENT PURCHASES. tHEY ARE FINANCIALLY WISE.*

*#----------*

*#BALANCE--> M*

*#PURCHASE FREQUENCY-->H*

*#CASH ADVANCE-->L*

*#INSTALLMENTS PURCHASES-->H*

*#CREDIT LIMIT-->VH*

*#PAYMENTS-->M*

*#FULL PAYMENT-->H*

##### *CLUSTER C4 PERSONA*

In [43]:

*#let´s characterize the Persona in the Cluster C4*

cluster\_C4=df[df['cluster']==4]

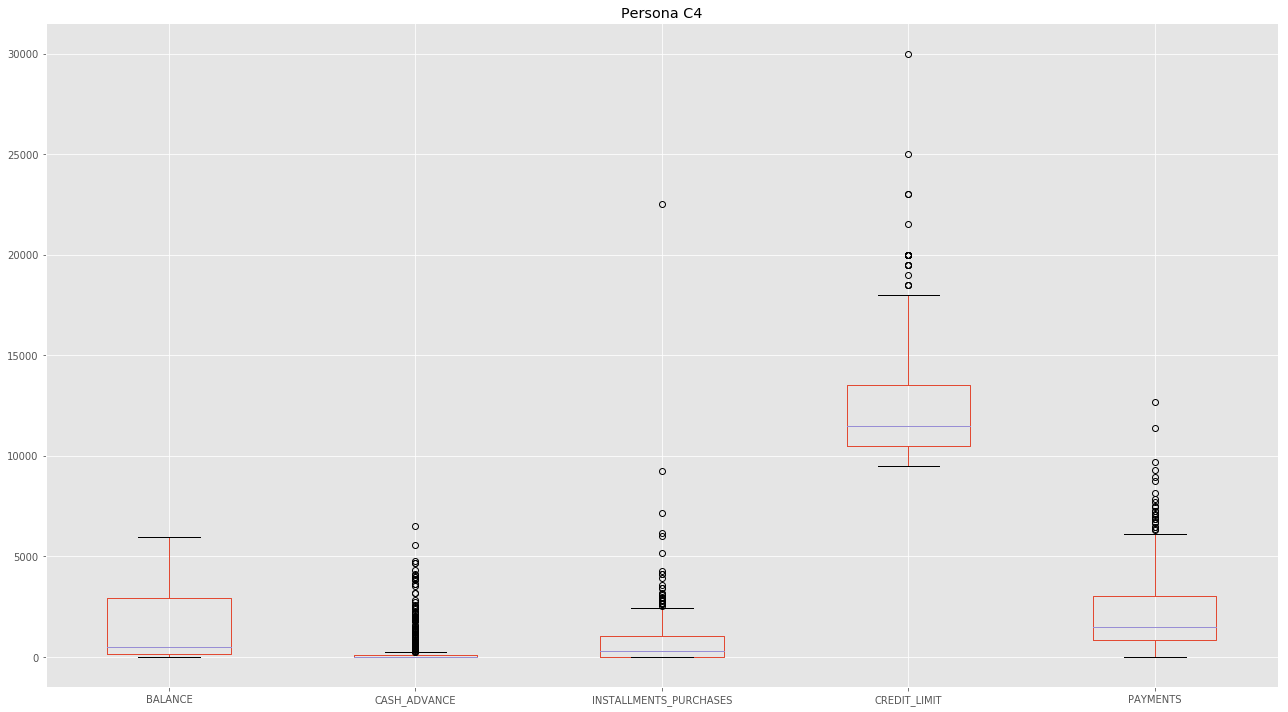
cluster\_C4[['BALANCE','CASH\_ADVANCE','INSTALLMENTS\_PURCHASES',

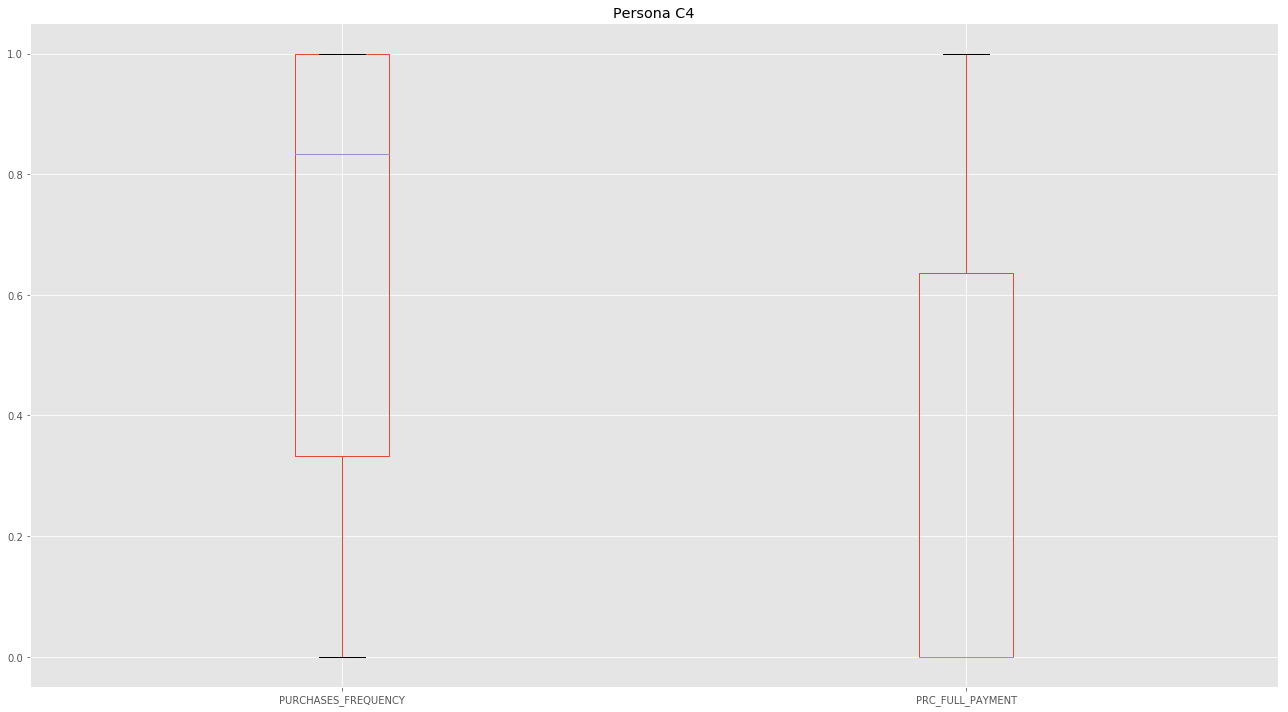
'CREDIT\_LIMIT','PAYMENTS']].plot.box(figsize=(18,10),by='cluster',title='Persona C4',legend=True);

plt.tight\_layout()

cluster\_C4[['PURCHASES\_FREQUENCY','PRC\_FULL\_PAYMENT']].plot.box(figsize=(18,10),title='Persona C4',legend=True);

plt.tight\_layout()

****

****

##### *CLUSTER C5*

In [44]:

*#CLUSTER C5-->457. INCREASE THE CREDIT LIMIT OF THESE CLIENTS*

*#----------*

*#BALANCE--> M*

*#PURCHASE FREQUENCY-->M*

*#CASH ADVANCE-->M*

*#INSTALLMENTS PURCHASES-->H*

*#CREDIT LIMIT-->H*

*#PAYMENTS-->H*

*#FULL PAYMENT-->H*

##### *CLUSTER C5 PERSONA*

In [45]:

*#let´s characterize the Persona in the Cluster C5*

cluster\_C5=df[df['cluster']==5]

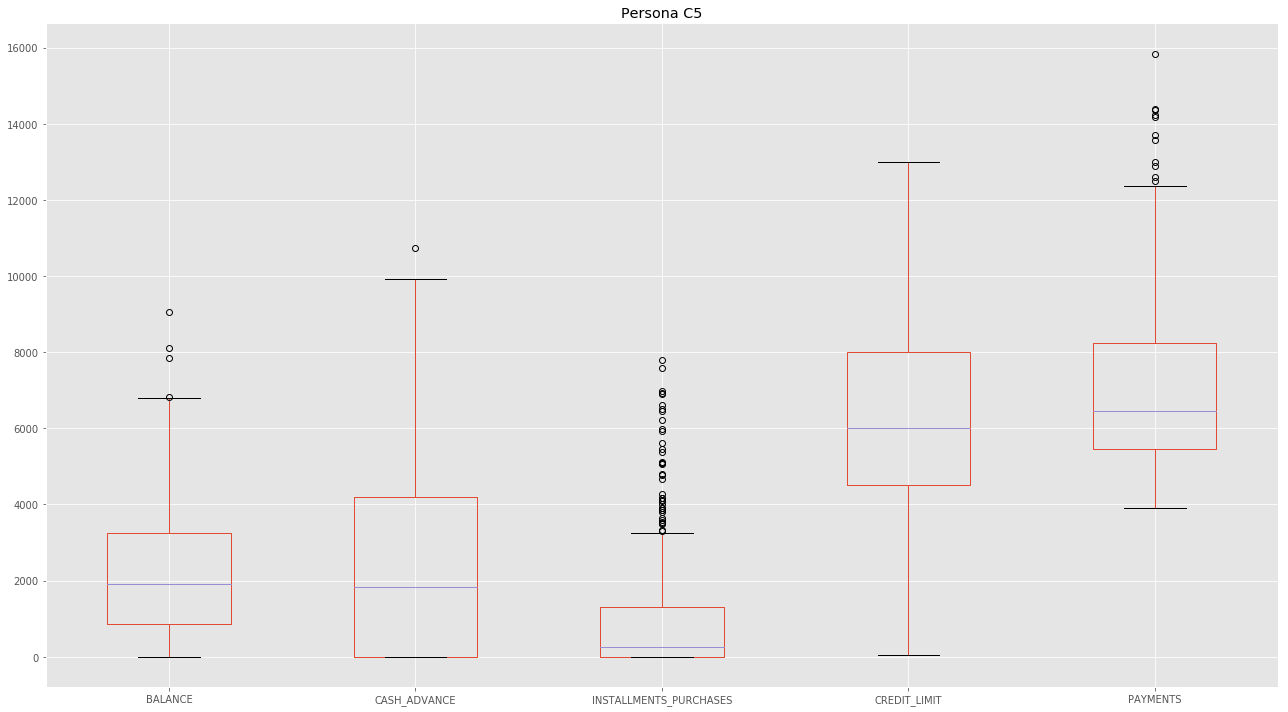
cluster\_C5[['BALANCE','CASH\_ADVANCE','INSTALLMENTS\_PURCHASES',

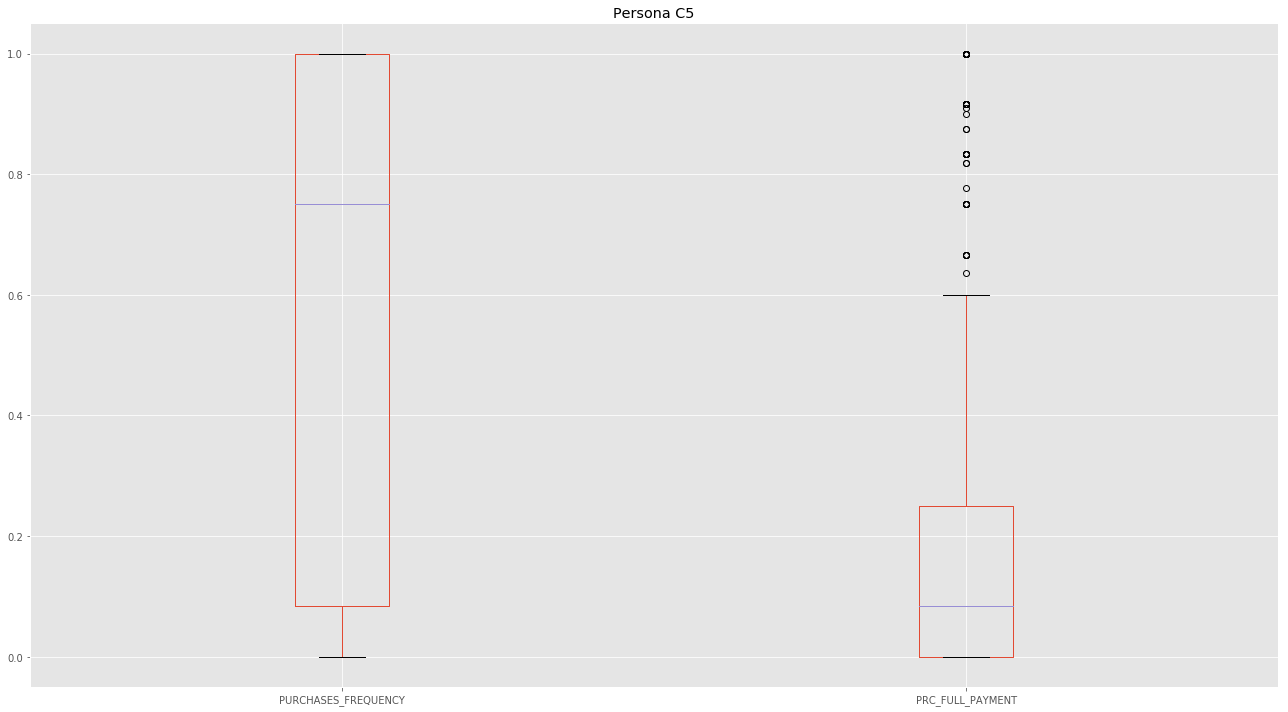
'CREDIT\_LIMIT','PAYMENTS']].plot.box(figsize=(18,10),by='cluster',title='Persona C5',legend=True);

plt.tight\_layout()

cluster\_C5[['PURCHASES\_FREQUENCY','PRC\_FULL\_PAYMENT']].plot.box(figsize=(18,10),title='Persona C5',legend=True);

plt.tight\_layout()

****

****

###### *CLUSTER C6*

In [46]:

*#CLUSTER C6-->443, VERY PROFITABLE CUSTOMERS. THEY USE THE CARD AS A REVOLVER.*

*#----------*

*#BALANCE--> H*

*#PURCHASE FREQUENCY-->L*

*#CASH ADVANCE-->M*

*#INSTALLMENTS PURCHASES-->L*

*#CREDIT LIMIT-->H*

*#PAYMENTS-->L*

*#FULL PAYMENT-->L*

##### *CLUSTER C6 PERSONA*

In [47]:

*#let´s characterize the Persona in the Cluster C6*

cluster\_C6=df[df['cluster']==2]

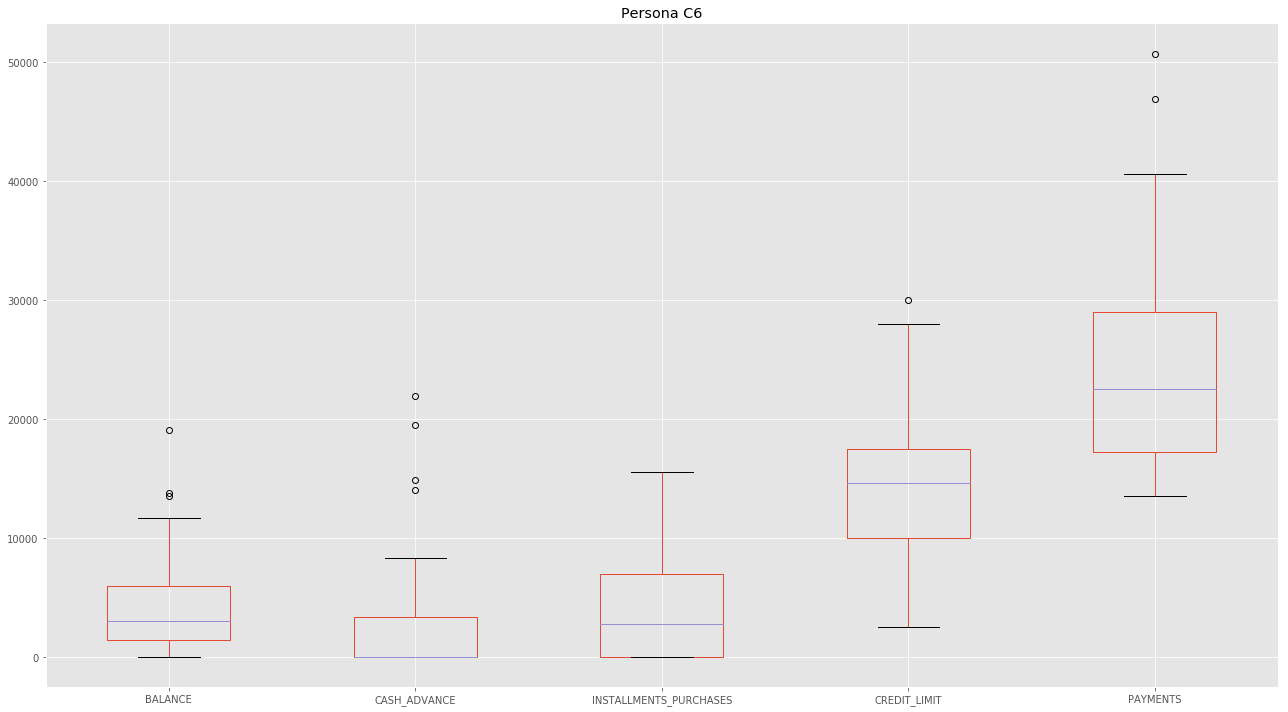
cluster\_C6[['BALANCE','CASH\_ADVANCE','INSTALLMENTS\_PURCHASES',

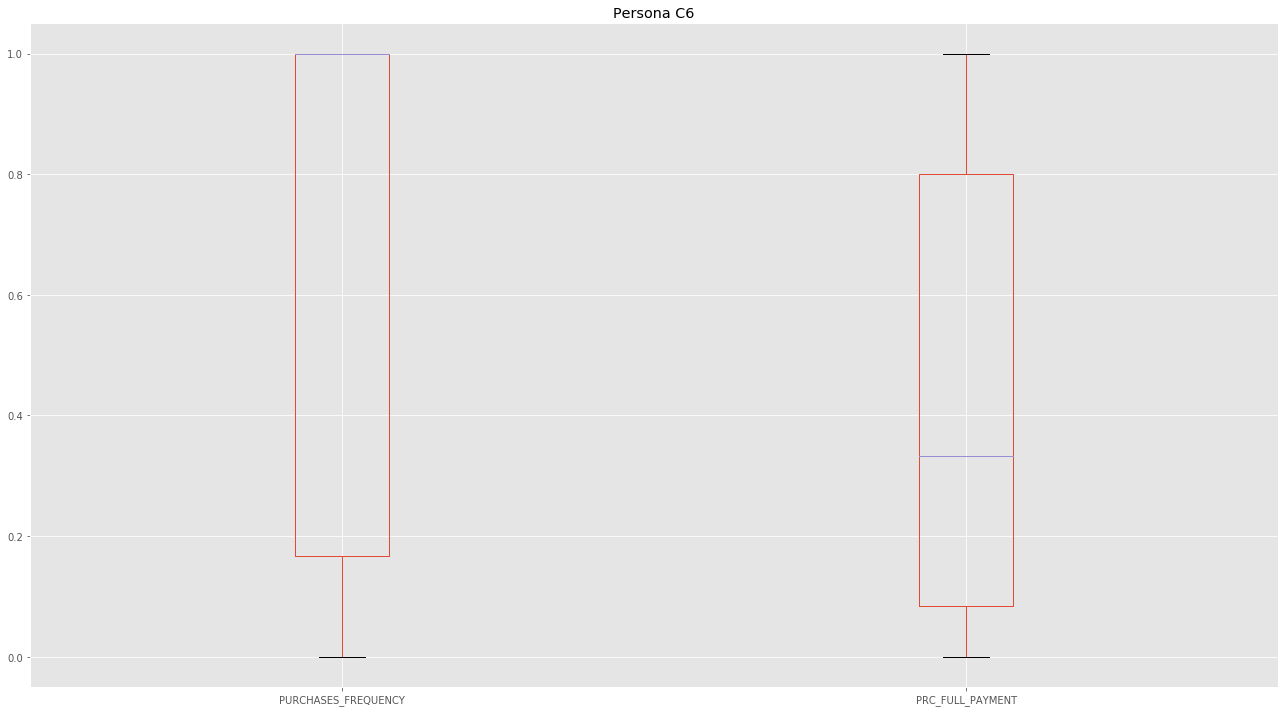
'CREDIT\_LIMIT','PAYMENTS']].plot.box(figsize=(18,10),by='cluster',title='Persona C6',legend=True);

plt.tight\_layout()

cluster\_C6[['PURCHASES\_FREQUENCY','PRC\_FULL\_PAYMENT']].plot.box(figsize=(18,10),title='Persona C6',legend=True);

plt.tight\_layout()

****

****

##### *CLUSTER C7*

In [48]:

*#CLUSTER C7-->89 THIS GROUP IS VERY PROFITABLE AS THE DRAW MONEY WITH THE CREDIT CARD AND THEY FORGET TO PAYBACK IT*

*#----------*

*#BALANCE--> H*

*#PURCHASE FREQUENCY-->L*

*#CASH ADVANCE-->VH*

*#INSTALLMENTS PURCHASES-->M*

*#CREDIT LIMIT-->VH*

*#PAYMENTS-->H*

*#FULL PAYMENT-->L*

##### *CLUSTER C7 PERSONA*

In [49]:

*#let´s characterize the Persona in the Cluster C7*

cluster\_C7=df[df['cluster']==7]

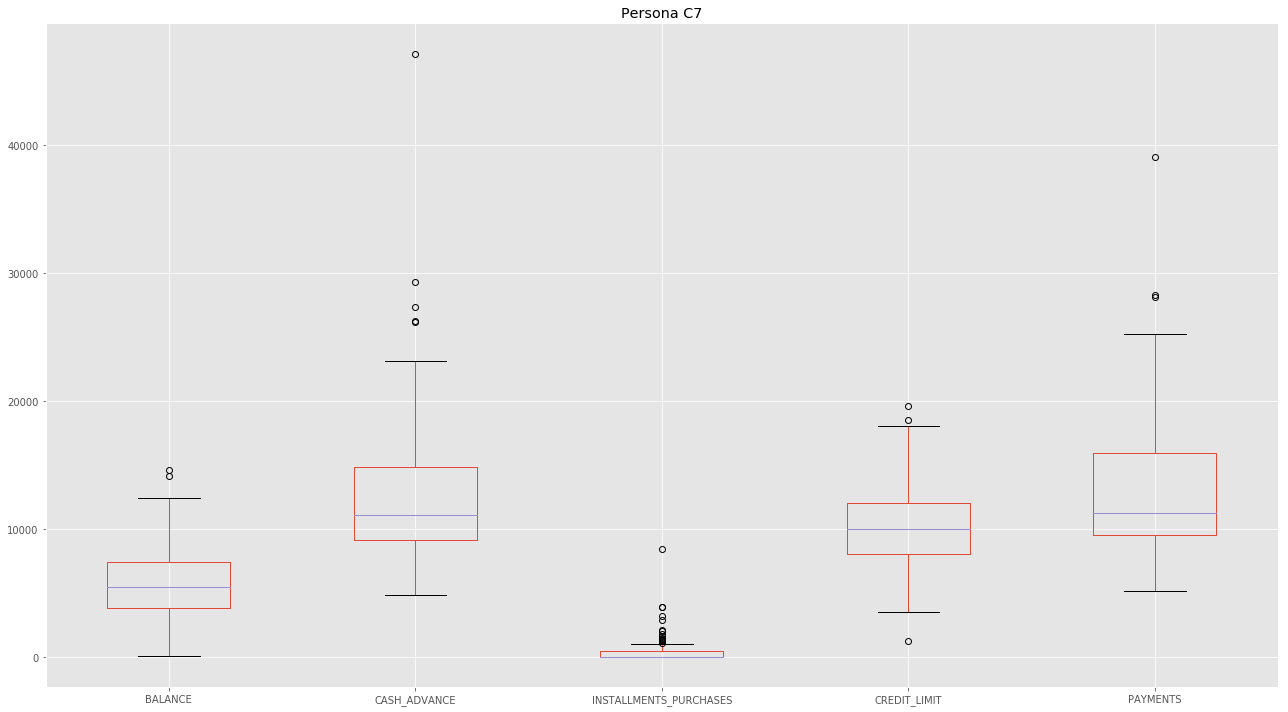
cluster\_C7[['BALANCE','CASH\_ADVANCE','INSTALLMENTS\_PURCHASES',

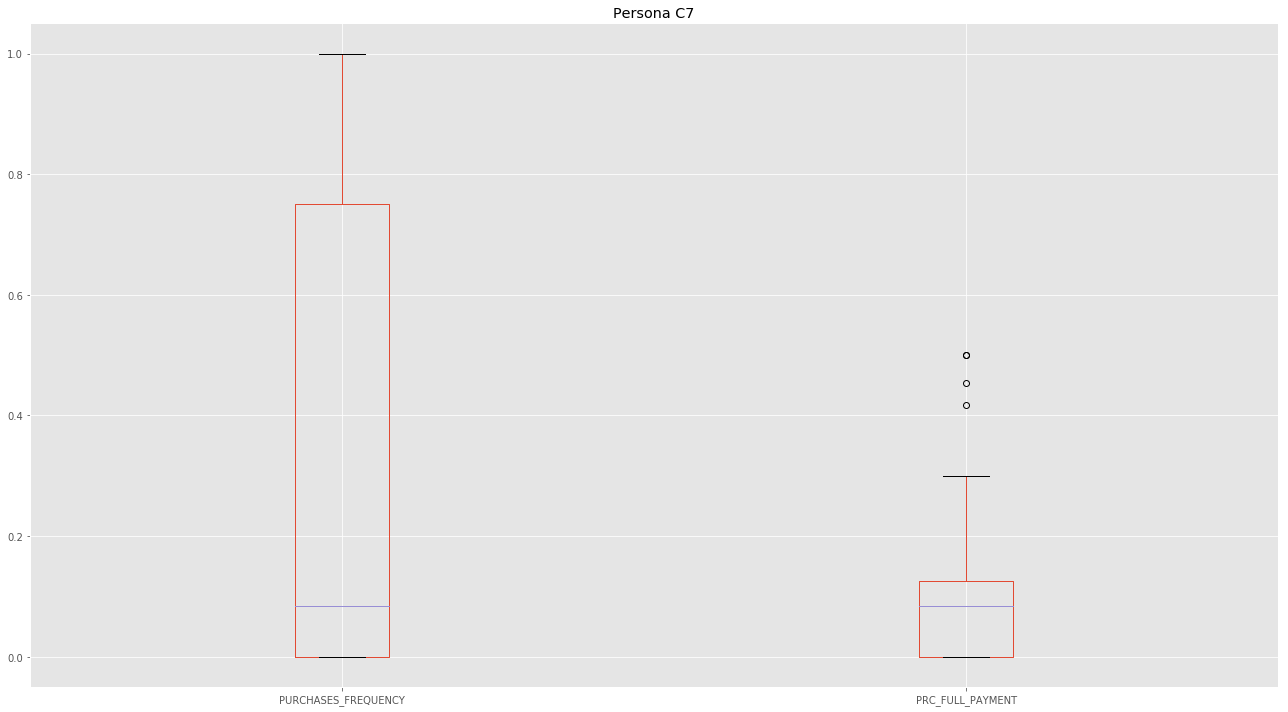
'CREDIT\_LIMIT','PAYMENTS']].plot.box(figsize=(18,10),by='cluster',title='Persona C7',legend=True);

plt.tight\_layout()

cluster\_C7[['PURCHASES\_FREQUENCY','PRC\_FULL\_PAYMENT']].plot.box(figsize=(18,10),title='Persona C7',legend=True);

plt.tight\_layout()

****

****

In [50]:

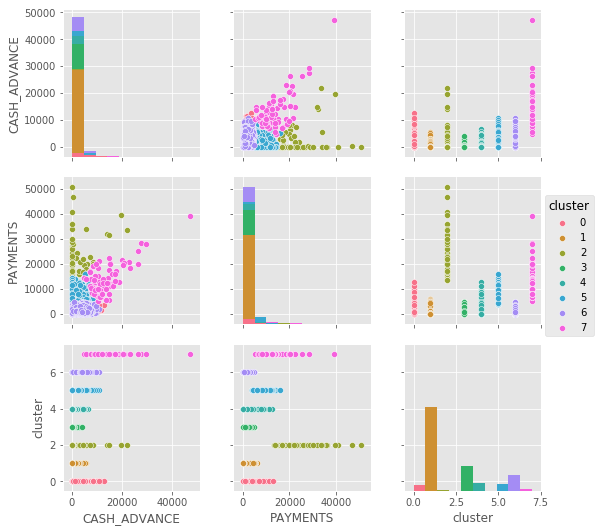
*#we can continue exploring the clusters in more detail*

best\_cols = ["CASH\_ADVANCE","PAYMENTS","cluster"]

fig=sns.pairplot( df[ best\_cols ], hue="cluster");

fig;

fig.savefig('clusters.png')

****

In [51]:

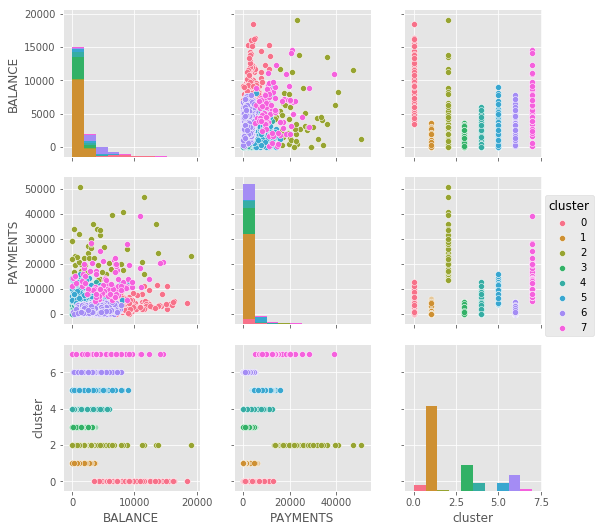
*#we can continue exploring the clusters in more detail*

best\_cols = ["BALANCE","PAYMENTS","cluster"]

fig=sns.pairplot( df[ best\_cols ], hue="cluster");

fig;

fig.savefig('clusters.png')

****

In [52]:

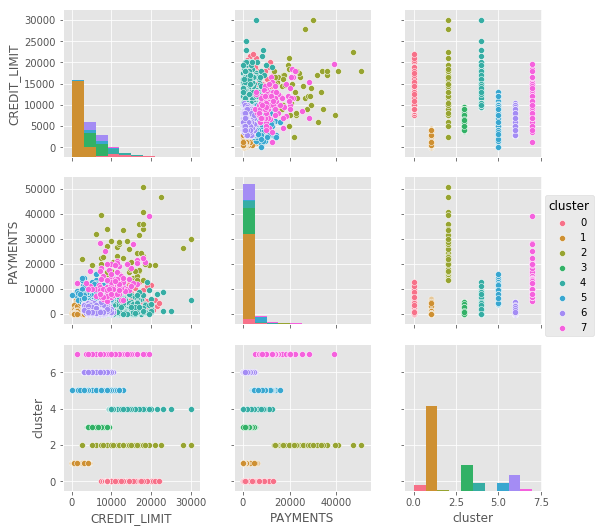
*#we can continue exploring the clusters in more detail*

best\_cols = ["CREDIT\_LIMIT","PAYMENTS","cluster"]

fig=sns.pairplot( df[ best\_cols ], hue="cluster");

fig;

fig.savefig('clusters.png')

****

In [53]:

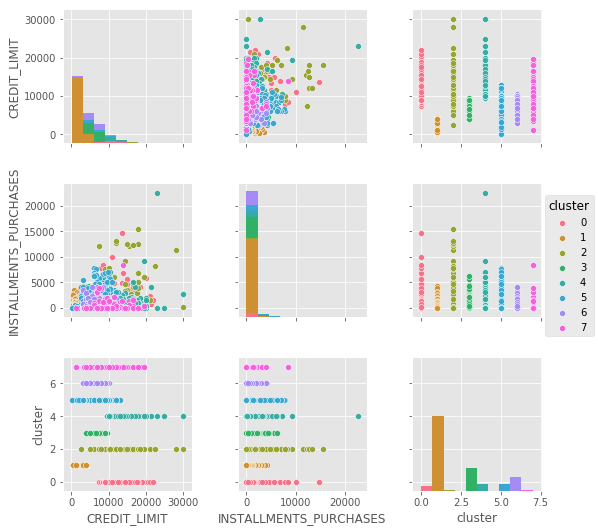
*#we can continue exploring the clusters in more detail*

best\_cols = ["CREDIT\_LIMIT","INSTALLMENTS\_PURCHASES","cluster"]

fig=sns.pairplot( df[ best\_cols ], hue="cluster");

fig;

fig.savefig('clusters.png')

****

In [54]:

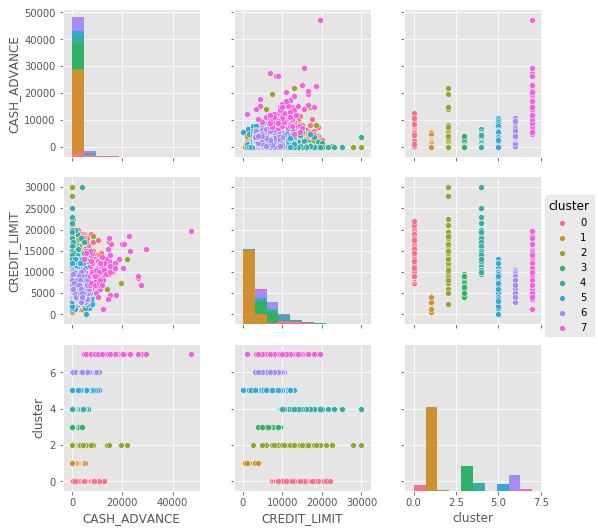
*#we can continue exploring the clusters in more detail*

best\_cols = ["CASH\_ADVANCE","CREDIT\_LIMIT","cluster"]

fig=sns.pairplot( df[ best\_cols ], hue="cluster");

fig;

fig.savefig('clusters.png')

****

In [ ]:

**R Code:**

|  |  |
| --- | --- |
| getwd() | |
|  | | setwd("/Users/apple/Desktop") | | |
|  | | file\_a <- read.csv(file.choose(),header = TRUE) | | |
|  | | file <- file\_a[,-1] | | |
|  | | View(file\_a) | | |
|  | | dim(file) | | |
|  | | colnames(file) | | |
|  | | str(file) | | |
|  | |  | | |
|  | | #Missing Values Deletion | | |
|  | | file$MINIMUM\_PAYMENTS[is.na(file$MINIMUM\_PAYMENTS)] <- 0 | | |
|  | | file$CREDIT\_LIMIT[is.na(file$CREDIT\_LIMIT)] <- 0 | | |
|  | | dim(file) | | |
|  | | str(file\_a) | | |
|  | | summary(file$CREDIT\_LIMIT) | | |
|  | | View(file) | | |
|  | |  | | |
|  | | #Creating New Variables & Advanced Data Preparation | | |
|  | | file$Monthly\_Avg\_expns <- file$PURCHASES/(file$PURCHASES\_FREQUENCY\*file$TENURE) | | |
|  | | file$Monthly\_CASH\_ADVANCE <- file$CASH\_ADVANCE/(file$CASH\_ADVANCE\_FREQUENCY\*file$TENURE) | | |
|  | | file$Monthly\_INSTALLMENTS\_PURCHASES <- file$INSTALLMENTS\_PURCHASES/(file$PURCHASES\_INSTALLMENTS\_FREQUENCY\*file$TENURE) | | |
|  | | file$Monthly\_ONEOFF\_PURCHASES <- file$ONEOFF\_PURCHASES/(file$ONEOFF\_PURCHASES\_FREQUENCY\*file$TENURE) | | |
|  | |  | | |
|  | | file$limit\_usage <- file$BALANCE/file$CREDIT\_LIMIT | | |
|  | | file$limit\_usage[file$limit\_usage==Inf] <- 0 | | |
|  | | file$ptmp <- file$PAYMENTS/file$MINIMUM\_PAYMENTS | | |
|  | | file$ptmp[file$ptmp==Inf] <- 0 | | |
|  | | file$no\_purchase <- file$PURCHASES/file$PURCHASES\_TRX | | |
|  | | file$Monthly\_Avg\_expns[is.nan(file$Monthly\_Avg\_expns)] <- 0 | | |
|  | | file$Monthly\_CASH\_ADVANCE[is.nan(file$Monthly\_CASH\_ADVANCE)] <- 0 | | |
|  | | file$Monthly\_INSTALLMENTS\_PURCHASES[is.nan(file$Monthly\_INSTALLMENTS\_PURCHASES)] <- 0 | | |
|  | | file$Monthly\_ONEOFF\_PURCHASES[is.nan(file$Monthly\_ONEOFF\_PURCHASES)] <- 0 | | |
|  | | file$ptmp[is.nan(file$ptmp)] <- 0 | | |
|  | | file$no\_purchase[is.nan(file$no\_purchase)] <- 0 | | |
|  | | file$no\_purchase[file$no\_purchase==Inf] <- 0 | | |
|  | | a <- as.factor(file$no\_purchase) | | |
|  | | A | | |
|  | | corrm <- cor(file) | | |
|  | | View(corrm) | | |
|  | | write.csv(corrm,"/Users/apple/Desktop/Credit\_card/corrm.csv") | | |
|  | |  | | |
|  | | #Data Preparation Outlier Treatment | | |
|  | | mystats <- function(x) { | | |
|  | | nmiss<-sum(is.na(x)) | | |
|  | | a <- x[!is.na(x)] | | |
|  | | med <- median(a) | | |
|  | | m <- mean(a) | | |
|  | | n <- length(a) | | |
|  | | s <- sd(a) | | |
|  | | min <- min(a) | | |
| p1<-quantile(a,0.01) | | |
| p5<-quantile(a,0.05) | | |
| p10<-quantile(a,0.10) | | |
| q1<-quantile(a,0.25) | | |
| q2<-quantile(a,0.5) | | |
| q3<-quantile(a,0.75) | | |
| p90<-quantile(a,0.90) | | |
| p95<-quantile(a,0.95) | | |
| p99<-quantile(a,0.99) | | |
| max <- max(a) | | |
| UC <- m+3\*s | | |
| LC <- m-3\*s | | |
| outlier\_flag<- max>UC | min<LC | | |
| return(c(n=n, nmiss=nmiss, outlier\_flag=outlier\_flag, median=med,mean=m, stdev=s,min = min, p1=p1,p5=p5,p10=p10,q1=q1,q2=q2,q3=q3,p90=p90,p95=p95,p99=p99,max=max, UC=UC, LC=LC )) | | |
| } | | |
|  | | |
| class(diag\_stats) | | |
| class(file) | | |
| diag\_stats<-t(data.frame(apply(file, 2, mystats))) | | |
| write.csv(diag\_stats, "diag\_stats.csv") | | |
| names(file) | | |
| diag\_stats <- as.data.frame(diag\_stats)diag\_stats$UC | | |
| file$BALANCE[file$BALANCE>7809.07046604776] <- 7809.07046604776 | | |
| file$PURCHASES[file$PURCHASES>7413.10917913822] <- 7413.10917913822 | | |
| file$INSTALLMENTS\_PURCHASES[file$INSTALLMENTS\_PURCHASES>3124.08199021888] <- 3124.08199021888 | | |
| file$CREDIT\_LIMIT[file$CREDIT\_LIMIT>15410.7149052053] <- 15410.7149052053 | | |
| file$CASH\_ADVANCE[file$CASH\_ADVANCE>9588.1633568] <- 9588.1633568 | | |
| file$PAYMENTS[file$PAYMENTS>10418.3351227384] <- 10418.3351227384 | | |
| file$MINIMUM\_PAYMENTS[file$MINIMUM\_PAYMENTS>7841.95320738377] <- 7841.95320738377 | | |
| file$Monthly\_Avg\_expns[file$Monthly\_Avg\_expns>986.029883028158] <- 986.029883028158 | | |
| file$Monthly\_CASH\_ADVANCE[file$Monthly\_CASH\_ADVANCE>3200.48675416176] <- 3200.48675416176 | | |
| file$Monthly\_INSTALLMENTS\_PURCHASES[file$Monthly\_INSTALLMENTS\_PURCHASES>439.463506363285] <- 439.463506363285 | | |
| file$Monthly\_ONEOFF\_PURCHASES[file$Monthly\_ONEOFF\_PURCHASES>1039.55862704017] <- 1039.55862704017 | | |
| file$ptmp[file$ptmp>363.554382806177] <- 363.554382806177 | | |
| file$no\_purchase[file$no\_purchase>555.44550033815] <- 555.44550033815 | | |
|  | | |
|  | | |
| #Factor Analysis | | |
| require(psych) | | |
| require(GPArotation) | | |
| scree(corrm, factors=T, pc=T, main="scree plot", hline=NULL, add=FALSE) ### SCREE PLOT | | |
| data.frame(eigen(corrm)$values ) | | |
| require(dplyr) | | |
| eigen\_values <- mutate(data.frame(eigen(corrm)$values) | | |
| ,cum\_sum\_eigen=cumsum(eigen.corrm..values) | | |
| , pct\_var=eigen.corrm..values/sum(eigen.corrm..values) | | |
|  | , cum\_pct\_var=cum\_sum\_eigen/sum(eigen.corrm..values)) # CALCULATING VARIANCE, CUMULATIVE VARIANCE etc... | | | |
|  | summary(file) | | | |
|  | FA<-fa(r=corrm, 8, rotate="varimax", fm="ml") ### CONDUCTING FACTOR ANALYSIS | | | |
|  | print(FA) ### PRINT THE RESULTS | | | |
|  | FA\_SORT<-fa.sort(FA) ### SORTING THE LOADINGS | | | |
|  | ls(FA\_SORT) ### LISTING OUT THE OBJECTS | | | |
|  | FA\_SORT$loadings | | | |
|  | #FA\_SORT$e.values ### FINDING EIGEN VALUES FROM THE RESULTS | | | |
|  | Loadings<-data.frame(FA\_SORT$loadings[1:ncol(file),]) ### CAPTURING ONLY LOADINGS INTO DATA FRAME | | | |
|  | write.csv(Loadings, "/Users/Desktop/Credit\_card/loadings1.csv") ### SAVING THE FILE | | | |
|  |  | | | |
|  | vars <- c("ONEOFF\_PURCHASES","PURCHASES","PAYMENTS","no\_purchase", | | | |
|  | "Monthly\_Avg\_expns","PURCHASES\_INSTALLMENTS\_FREQUENCY","PURCHASES\_FREQUENCY","PURCHASES\_TRX", | | | |
|  | "INSTALLMENTS\_PURCHASES","BALANCE","CASH\_ADVANCE\_FREQUENCY","ONEOFF\_PURCHASES\_FREQUENCY") | | | |
|  | inputdata\_final <-file[vars] | | | |
|  |  | | | |
|  | #Prepare final Data | | | |
|  | #standardizing the data | | | |
|  | inputdata\_final = scale(inputdata\_final) | | | |
|  | View(inputdata\_final) | | | |
|  | #View(inputdata\_final) | | | |
|  | #building clusters using k-means clustering | | | |
|  | cluster\_three <- kmeans(inputdata\_final,3) | | | |
|  | cluster\_four <- kmeans(inputdata\_final,4) | | | |
|  | cluster\_five <- kmeans(inputdata\_final,5) | | | |
|  | cluster\_six <- kmeans(inputdata\_final,6) | | | |
|  |  | | | |
|  | file\_new<-cbind(file,km\_clust\_3=cluster\_three$cluster,km\_clust\_4=cluster\_four$cluster, | | | |
|  | km\_clust\_5=cluster\_five$cluster ,km\_clust\_6=cluster\_six$cluster ) | | | |
|  | View(file\_new) | | | |
|  |  | | | |
|  | #Graph based on k-means - Optional | | | |
|  | require(cluster) | | | |
|  |  | | | |
|  | clusplot(inputdata\_final, #dataframe | | | |
|  | cluster\_five$cluster, #clusterdata | | | |
|  | color = TRUE, #color | | | |
|  | #shade = TRUE, # Lines in clusters | | | |
|  | lines =6, # lines connecting centroids | | | |
|  | labels = 2 # Labels clusters and cases | | | |
|  | ) | | | |
|  |  | | | |
|  | ###Profiling | | | |
|  |  | | | |
|  | require(tables) | | | |
| tt<-cbind(tt<-cbind(tabular(1+factor(km\_clust\_3)+factor(km\_clust\_4)+factor(km\_clust\_5)+factor(km\_clust\_6) | | | |
| ~ Heading()\*length\*All(file[1]), data=file\_new)), | | | |
| tabular( 1+factor(km\_clust\_3)+factor(km\_clust\_4)+factor(km\_clust\_5)+factor(km\_clust\_6) | | | |
| ~ Heading()\*mean\*All(file[vars]), data=file\_new)) | | | |
| tt1<-as.data.frame.matrix(tt) | | | |
| #View(tt1) | | | |
|  | | | |
| rownames(tt1)<-c("Overall", "KM3\_1" ,"KM3\_2" ,"KM3\_3", "KM4\_1" ,"KM4\_2" ,"KM4\_3", "KM4\_4" ,"KM5\_1" ,"KM5\_2", "KM5\_3" ,"KM5\_4" ,"KM5\_5", "KM6\_1" ,"KM6\_2" ,"KM6\_3", "KM6\_4" ,"KM6\_5" ,"KM6\_6") | | | |
| colnames(tt1)<-c("Count","ONEOFF\_PURCHASES","PURCHASES","PAYMENTS","no\_purchase", | | | |
| "Monthly\_Avg\_expns","PURCHASES\_INSTALLMENTS\_FREQUENCY","PURCHASES\_FREQUENCY","PURCHASES\_TRX", | | | |
| "INSTALLMENTS\_PURCHASES","BALANCE","CASH\_ADVANCE\_FREQUENCY","ONEOFF\_PURCHASES\_FREQUENCY") | | | |
| cluster\_profiling<-t(tt1) | | | |
|  | | | |
| write.csv(cluster\_profiling, "/Desktop/Credit\_card/cluster\_profiling\_withoutoutliers.csv") | | | |
|  | | | |

R and Python files:



