Projekt końcowy Data Science

Dorota Gawrońska-Popa

Klasteryzacja danych

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree


from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler


from sklearn.decomposition import PCA
from sklearn.decomposition import IncrementalPCA

from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
```

Klasteryzacja bedzie robiona na zbiorze data_dummies

In [2]:
```
# wczytuje dane z poprzedniej części
data_dummies = pd.read_csv('data_dummies.csv')
```

In [3]:
```
data_dummies.head()
```

Out[3]:

	Unnamed: 0	loan_amnt	funded_amnt	term	int_rate	installment	sub_grade	emp_length
0	0	5000.0	5000.0	1	10.65	162.87	1	10
1	1	2500.0	2500.0	2	15.27	59.83	2	1
2	2	2400.0	2400.0	1	15.96	84.33	3	10
3	3	10000.0	10000.0	1	13.49	339.31	4	10
4	4	3000.0	3000.0	2	12.69	67.79	5	1

5 rows × 99 columns

In [4]:
```
data_dummies.drop(['loan_status', 'Unnamed: 0'], axis=1, inplace=Tr
```

In [5]:
```
data_dummies.shape
```

Out[5]: (42535, 97)

In [6]:
```
data_dummies.head()
```

Out[6]:

	loan_amnt	funded_amnt	term	int_rate	installment	sub_grade	emp_length	annual_inc
0	5000.0	5000.0	1	10.65	162.87	1	10	24000.0
1	2500.0	2500.0	2	15.27	59.83	2	1	30000.0
2	2400.0	2400.0	1	15.96	84.33	3	10	12252.0
3	10000.0	10000.0	1	13.49	339.31	4	10	49200.0
4	3000.0	3000.0	2	12.69	67.79	5	1	80000.0

5 rows × 97 columns

In [7]:
```
scaler = StandardScaler()
scaler.fit(data_dummies)
data_dummies_std = scaler.transform(data_dummies)
```

Przygotowanie zbioru do grupowania z PCA

- Zmniejszenie liczby wymiarów.
- Usunięcie wartości ujemnych.
- Usunięcie skośności zmiennych.
- Usuwanie ouliersów.
- Skalowanie danych.

Zmniejszenie liczby wymiarów

In [8]:
```
pca = IncrementalPCA(n_components=2)
data_dummies_pca = pca.fit_transform(data_dummies)
data_pca = pd.DataFrame(data_dummies_pca, columns = ['c1', 'c2'], i
```

In [9]:
```
data_pca
```

Out[9]:

	c1	c2
0	-45282.271425	3158.565432
1	-40786.620453	-9829.325366
2	-58279.298352	-6572.469319
3	-20803.384098	-6458.204225
4	11728.517628	10115.675027
...	...	...
42530	108047.180124	-28463.034380
42531	-58942.348408	-9764.908250
42532	38472.583043	-20656.666720
42533	-10916.473014	-13995.292702
42534	-1084.998316	-15491.374954

42535 rows × 2 columns

In [10]:
```
colnames = list(data_dummies.columns)
data_pca1 = pd.DataFrame({'c1':pca.components_[0],'c2':pca.componen
```

In [11]:
```
data_pca1
```

Out[11]:

	c1	c2	Feature
0	3.368171e-02	1.186367e-01	loan_amnt
1	3.202302e-02	1.136052e-01	funded_amnt
2	3.275048e-07	1.681826e-06	term
3	3.460200e-06	2.094485e-05	int_rate
4	9.556052e-04	3.336930e-03	installment
...	...	...	...
92	-5.197144e-09	-1.065550e-08	addr_state_VT
93	-1.175526e-08	6.883287e-08	addr_state_WA
94	-1.736147e-08	2.939460e-08	addr_state_WI
95	-1.603709e-08	-1.764391e-08	addr_state_WV
96	-4.312381e-09	6.521946e-09	addr_state_WY

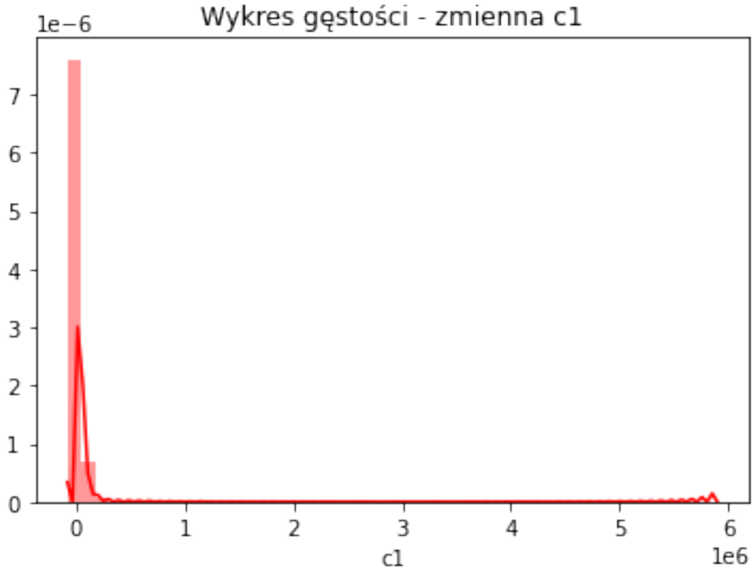
97 rows × 3 columns

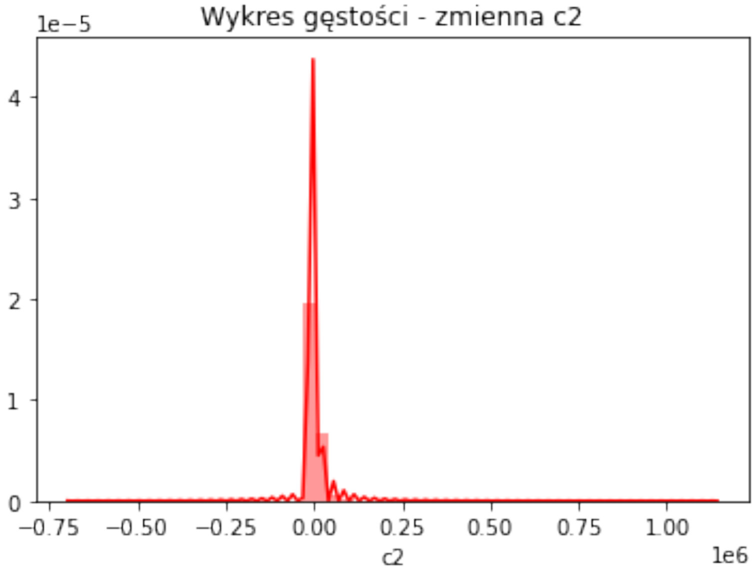
In [12]:
```
data_pca.agg(['mean','median','std','min','max']).round(2)
```

Out[12]:

	c1	c2
mean	0.00	-0.00
median	-10743.68	-4325.41
std	64499.50	21068.50
min	-68905.31	-694589.37
max	5887602.48	1139657.99

In [13]:
```
sns.distplot(data_pca.c1, color = 'r').set(title = 'Wykres gęstości
plt.show()
sns.distplot(data_pca.c2, color = 'r').set(title = 'Wykres gęstości
plt.show()
```





Usuwanie wartości ujemnych, przez dodanie modułu z min najmniejszej wartości i plus 1

In [14]:
```
for col in data_pca:
    if data_pca[col].min() <= 0:
        data_pca[col] = data_pca[col] + np.abs(data_pca[col].min())
```

Usuwanie skośności

In [15]:
```
data_pca = np.log(data_pca)
```

Usuwanie wartości odstających

In [16]:
```
q1 = data_pca.quantile(0.25)
q3 = data_pca.quantile(0.75)
iqr = q3 - q1

low_boundary = (q1 - 1.5 * iqr)
upp_boundary = (q3 + 1.5 * iqr)
num_of_outliers_L = (data_pca[iqr.index] < low_boundary).sum()
num_of_outliers_U = (data_pca[iqr.index] > upp_boundary).sum()
outliers = pd.DataFrame({'lower_boundary':low_boundary, 'upper_boun
                         'num_of_outliers__lower_boundary':num_of_o
                         'num_of_outliers__upper_boundary':num_of_o
```

In [17]:
```
outliers
```

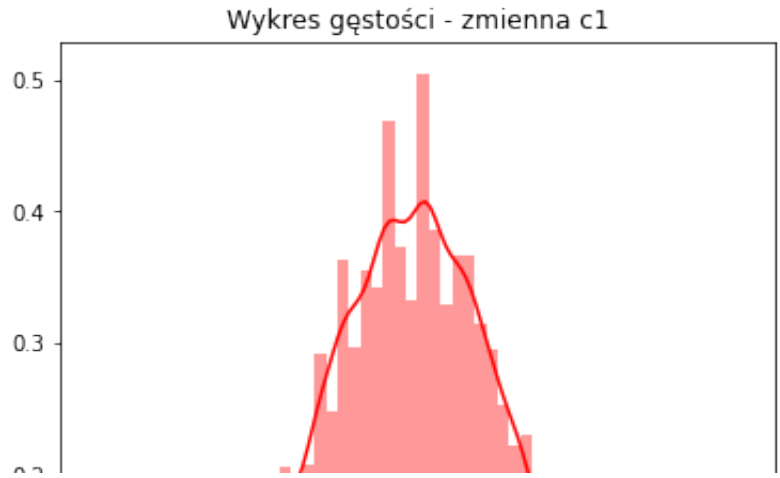
Out[17]:

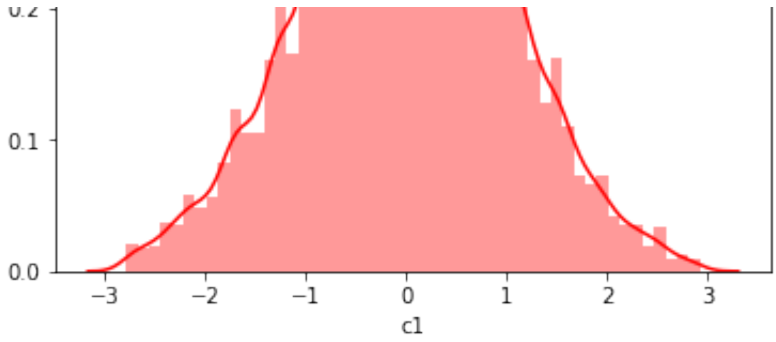
	lower_boundary	upper_boundary	num_of_outliers__lower_boundary	num_of_outliers__up
c1	9.492173	12.425370	521	
c2	13.409988	13.483962	188	

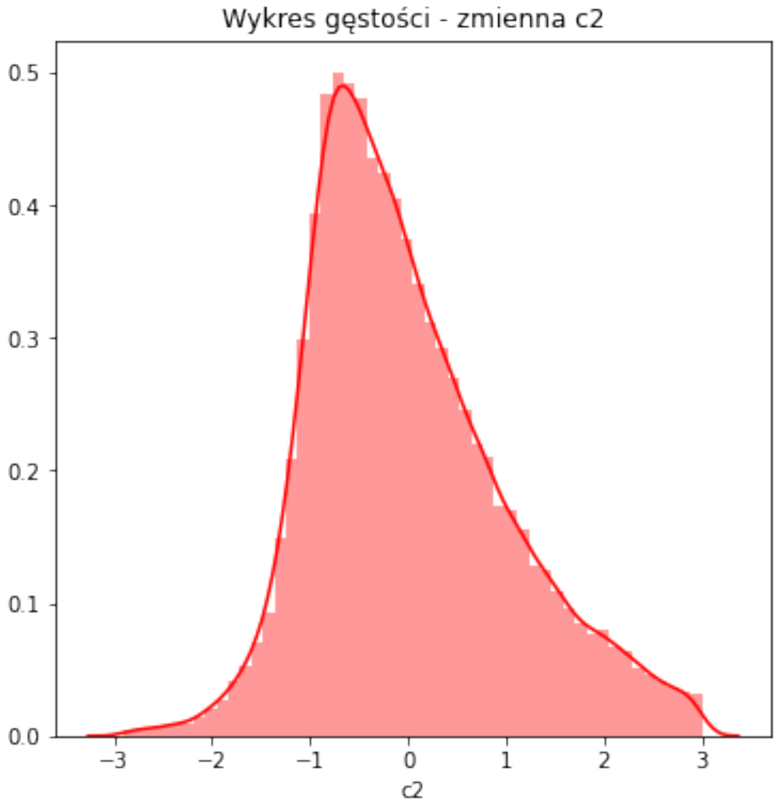
In [18]:
```
r row in outliers.iterrows():
  data_pca = data_pca[(data_pca[row[0]] >= row[1]['lower_boundary'])
```

In [19]:
```
scaler = StandardScaler()
scaler.fit(data_pca)
data_pca_std = scaler.transform(data_pca)
data_pca = pd.DataFrame(data=data_pca_std, index=data_pca.index, co
```

In [20]:
```
plt.figure(figsize = (6,6))
sns.distplot(data_pca.c1, color = 'r').set(title = 'Wykres gęstości
plt.show()
plt.figure(figsize = (6,6))
sns.distplot(data_pca.c2, color = 'r').set(title = 'Wykres gęstości
plt.show()
```





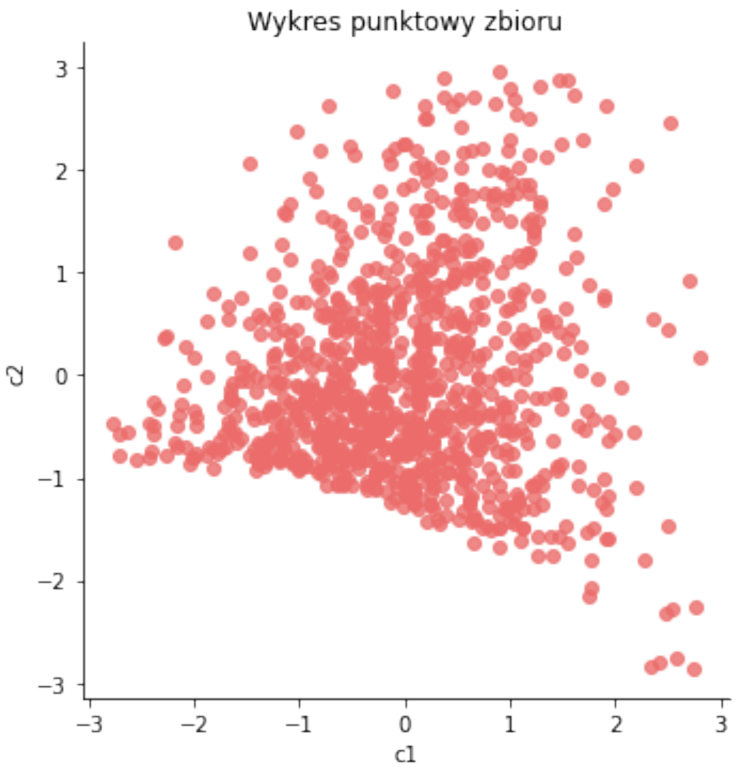


Jako, że hierarchiczna metoda klasteryzacji jest złożona obliczeniowo, ograniczam liczbę obserwacji do 1000. Wcześniej eksportuję plik po pCA do dalszej obróbki w notebooku modelowaniA.

In [21]:
```
# ten plik ma 1000 próbek
data_pca = data_pca.sample(1000, random_state = 67)
```

In [22]:
```
plt.figure(figsize = (20,20))
sns.lmplot('c1', 'c2', data = data_pca, scatter_kws={"color": "#eb6
plt.show()
```

```
<Figure size 1440x1440 with 0 Axes>
```



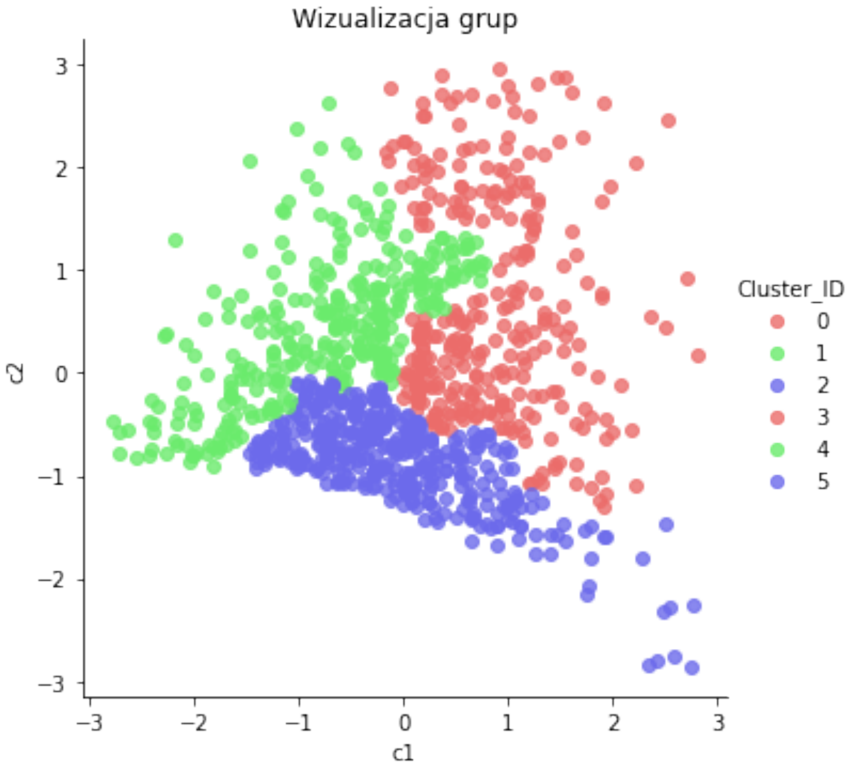
Klastrowanie hierarchiczne metodą Warda

Metoda łączenia grup Warda w bibliotece sklearn.

In [23]:
```
model_skl = AgglomerativeClustering(linkage='ward', affinity='eucli
```

In [24]:
```
data_pca['Cluster_ID'] = model_skl.labels_
```

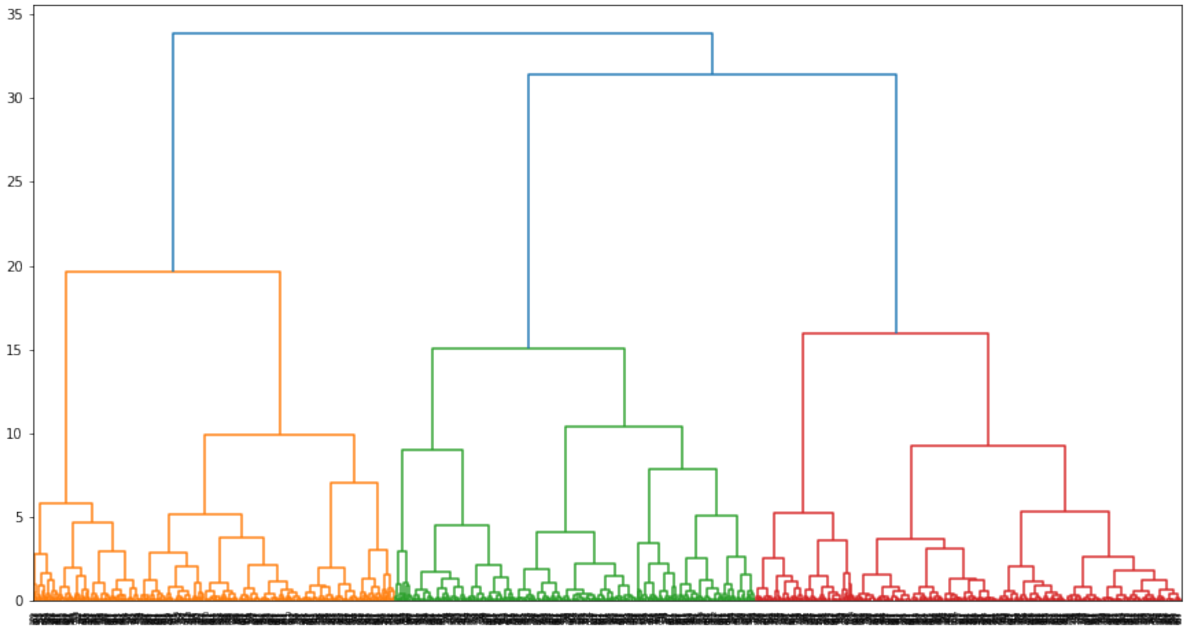
In [25]:
```
'c2', data = data_pca, hue = 'Cluster_ID', fit_reg=False, palette =
```



Grupowanie z użyciem biblioteki SciPy

In [26]:
```
model_sci = linkage(data_pca.iloc[:,0:2], method = 'ward', metric =
```

In [27]:
```
fig = plt.figure(figsize=(15, 8))
dn = dendrogram(model_sci)
plt.show()
```



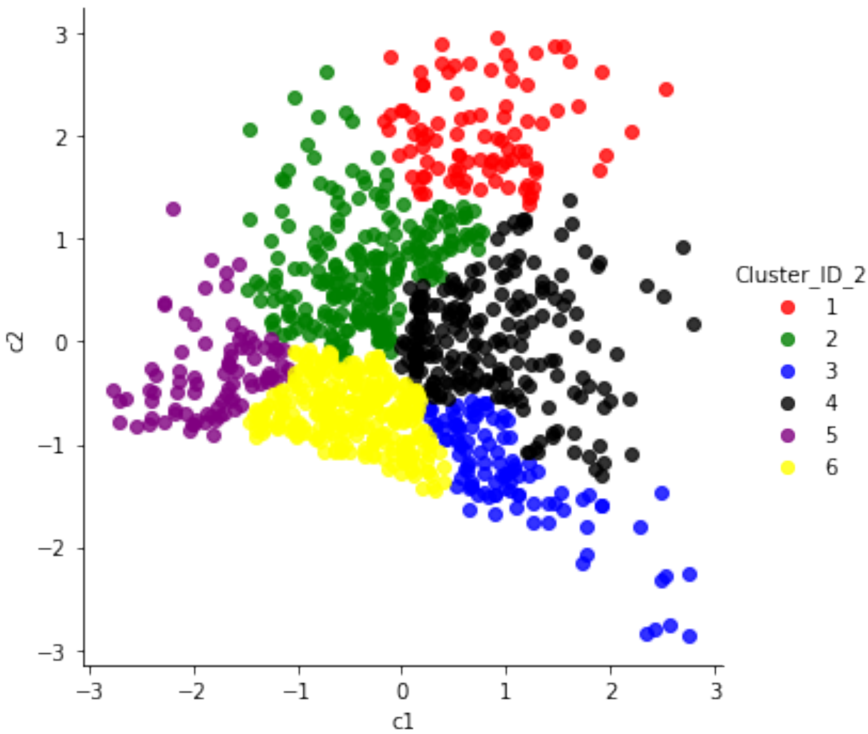
Dzielenie obserwacji na grupy według maksymalnej liczby klastrów.

In [28]:
```
clusters = fcluster(model_sci, 6, criterion='maxclust')
data_pca['Cluster_ID_2'] = clusters
clusters
```

Out[28]:
```
array([2, 4, 4, 1, 5, 4, 2, 6, 2, 2, 1, 3, 6, 3, 6, 4, 4, 2, 4, 6,
2, 6,
       4, 5, 4, 6, 6, 1, 2, 6, 5, 2, 5, 4, 4, 6, 3, 5, 4, 2, 6, 3,
3, 5,
       2, 2, 4, 4, 2, 4, 4, 2, 6, 3, 6, 1, 5, 2, 5, 6, 4, 6, 6, 4,
3, 2,
       6, 2, 5, 2, 5, 4, 6, 3, 5, 3, 4, 4, 2, 2, 1, 6, 2, 1, 6, 4,
2, 5,
       2, 2, 3, 3, 6, 4, 5, 6, 6, 4, 6, 6, 2, 2, 2, 6, 4, 6, 2, 3,
6, 6,
       5, 4, 4, 2, 4, 2, 5, 3, 2, 2, 2, 1, 4, 4, 3, 6, 5, 4, 3, 2,
5, 4,
       6, 4, 6, 4, 6, 2, 6, 5, 6, 6, 3, 2, 5, 2, 5, 2, 2, 4, 1, 4,
3, 4,
       3, 6, 1, 5, 5, 3, 1, 2, 4, 6, 2, 6, 4, 4, 3, 6, 4, 2, 1, 4,
5, 6,
       2, 3, 6, 2, 4, 3, 6, 6, 4, 4, 1, 6, 5, 2, 2, 4, 4, 3, 3, 2,
4, 2,
       5, 5, 3, 1, 6, 6, 6, 6, 6, 2, 2, 1, 6, 6, 4, 4, 2, 5, 6, 2,
```

In [29]:
```
plt.figure(figsize = (20,20))
sns.lmplot('c1', 'c2', data = data_pca, fit_reg=False, hue = 'Clust
plt.show()
```

<Figure size 1440x1440 with 0 Axes>



W obu metodach są takie same wyniki.

DBSCAN

Na pliku data_pca po PCA

In [30]:
```
clt = DBSCAN(eps=0.5, metric='euclidean', min_samples=5, n_jobs=-1)
model = clt.fit(data_pca)
```

In [31]:
```
data_pca['Clasters'] = model.labels_
```

In [32]:
```
model.labels_
```

Out[32]:
```
array([ 0,  1,  1,  2,  3,  1,  0,  4,  0,  0,  2,  5,  4,  5,  4,
1,  1,
        0,  1,  4,  0,  4,  1,  3,  1,  4,  4,  2,  0,  4,  3,  0,
3,  1,
        1,  4,  5,  3,  1,  0,  4,  5,  5,  3,  0,  0,  1,  1,  0,
1,  1,
        0,  4,  5,  4,  2,  3,  0,  3,  4,  1,  4,  4,  1,  5,  0,
4,  0,
        3,  0,  3, -1,  4,  5,  3,  6,  1,  1,  0,  0,  2,  4,  0,
2.  4.
```

```
 ,    ,
       -1,  0,  3,  0,  0,  5,  5,  4,  1,  3,  4,  4,  1,  4,  4,
 0,  0,
        0,  4,  1,  4,  0,  5,  4,  4,  3,  1,  1,  0,  1,  0,  3,
 5,  0,
        0,  0,  2,  1,  1,  5,  4,  3,  1,  5,  0,  3,  1,  4,  1,
 4,  1,
        4,  0,  4,  3,  4,  4,  5,  0,  3,  0,  3,  0,  0,  1,  2,
 1,  5,
        1,  5,  4,  2,  3,  3,  6,  2,  0,  1,  4,  0,  4,  1,  1,
 5,  4,
        1,  0,  2,  1,  3,  4,  0,  5,  4,  0,  1,  5,  4,  4,  1,
 1,  2,
        4,  3,  0,  0,  1,  1,  5,  5,  0,  1,  0,  3,  3,  5,  2,
 4,  4,
        4,  4,  4,  0,  0,  2,  4,  4,  1,  1,  0,  3,  4,  0,  5,
 4,  3,
        4,  1,  1,  0,  0,  0,  3,  2,  0,  4,  4,  2,  4,  5,  4,
 1,  1,
        3,  1,  0,  2,  4,  1,  4,  5,  5,  4, -1,  4,  0,  4,  1,
 4,  4,
        2,  4,  1,  2,  4,  0,  5,  1,  6,  4,  5,  5,  4,  1,  4,
 3,  4,
        0,  3,  5,  3,  4,  4,  1,  0,  0,  1,  4,  0,  0,  4,  3,
 1,  0,
        3,  5,  3,  5,  4,  2,  1,  0,  4,  5,  1,  1,  1,  0,  4,
 3,  4,
        4,  2,  0,  4,  6,  5,  4,  3,  5,  3,  2,  4,  5,  0,  2,
 1,  0,
        4,  1,  4,  2,  5,  4,  5,  0,  1,  1,  1,  4,  1,  4,  2,
 4,  0,
        1,  3,  0,  4,  4,  1,  1,  0,  4,  4,  4,  3,  3,  5,  4,
 4,  1,
        4,  1,  4,  4,  4,  1,  4,  4,  4,  1,  4,  4,  1,  1,  2,
 0,  0,
        3,  0,  1,  0,  1,  1,  0,  1,  4,  2,  0,  4,  5,  0,  1,
 1,  5,
        5,  0,  4,  1,  4,  2,  1,  0,  4,  5,  2,  4,  1,  1,  4,
 3, -1,
        2,  0,  0,  4, -1,  3,  5,  0,  1,  4,  0,  1,  2,  5,  2,
 4,  1,
        2,  0,  2,  0,  0,  1,  5,  5,  2,  1,  4,  4,  0,  0,  1,
 4,  1,
        5,  5,  5,  4,  0,  4,  4,  1,  0,  4,  1,  0,  0,  2,  0,
 1,  0,
        1, -1,  1,  4,  3,  0,  4,  5,  2,  0,  4,  0,  5,  3,  0,
 4,  4,
        5,  1,  4,  3,  0,  2,  1,  0,  0,  4,  4,  4,  4,  4,  1,
 4,  0,
        5,  2,  0,  4,  1,  0,  4,  2,  1,  0,  0,  1,  4,  1,  1,
 0,  4,
        2,  4,  4,  3,  0,  2,  0,  0,  3,  0,  1,  1,  4,  3,  2,
 1,  4,
        4,  1,  5,  4,  4,  3,  1,  1,  1,  2,  4,  5,  4,  2,  4,
```

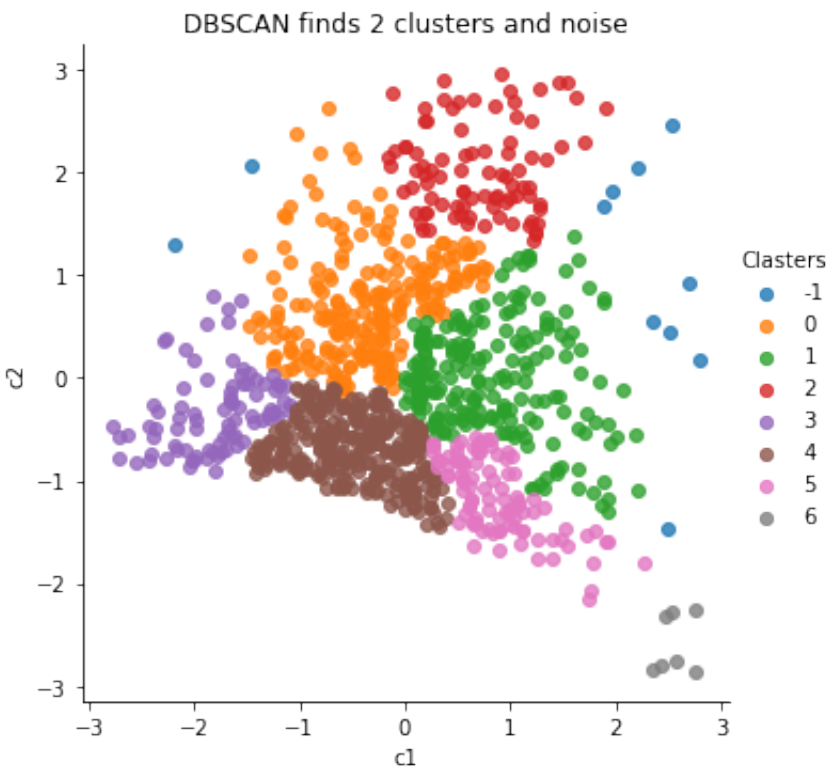
```
       1,  4,
               5,  0,  3,  2,  0,  5,  4,  2,  0,  2,  1,  4,  2,  5,  1,
       1,  5,
               2,  4,  3,  2,  1,  1, -1,  0,  1,  4,  2,  1,  4,  1,  5,
      -1,  0,
               1,  4,  1,  1,  5,  2,  1,  2,  1,  1,  6,  4,  0,  0,  0,
       0,  0,
               5,  1,  5,  5,  4,  1,  0,  1,  1,  4,  4,  2,  4,  3,  1,
       0,  5,
               5,  4,  4,  4,  1,  4,  4,  1,  2,  5,  3,  2,  3,  3,  0,
       0,  0,
               0,  0,  0,  3,  4,  4,  0,  0,  4,  3,  4,  0,  0,  1,  4,
       5,  5,
               0,  1,  1,  2,  2,  2,  4,  0,  4,  0,  0,  4,  5,  4,  4,
       3,  4,
               5,  4,  4,  1,  1,  3,  0,  3,  1,  4,  1,  0,  1,  5,  5,
       0,  4,
               1,  0,  4,  4,  4,  0,  3,  4,  4,  0,  1,  0,  0,  4,  1,
       3,  4,
               4,  4,  4,  3,  4,  4,  2,  4,  1,  3,  4,  4,  4,  4,  1,
       1,  4,
               2,  4,  4,  0,  0,  1,  1,  3,  4,  5,  4,  1,  1,  4,  3,
       4,  1,
               0,  4,  4,  1,  0,  0,  0,  0,  0,  0,  4,  4,  4,  4,  1,
       4,  5,
               2,  1,  2,  0,  5,  0,  4,  2,  3,  1,  1,  5,  0,  0,  5,
       2,  6,
               0,  1,  0,  0,  1,  1,  4,  3, -1,  2,  4,  4,  4,  3,  6,
       5,  0,
               5,  0,  0,  5,  1,  4,  4,  2,  1,  0,  4,  2,  4,  4,  4,
       4,  0,
               0,  4,  4,  4,  4,  4,  0,  1,  0,  0,  5,  0,  4,  3,  5,
       5,  0,
               2,  0,  4,  5,  0,  5,  4,  1,  4,  0,  4,  4,  4,  5,  3,
       4,  1,
               0,  4,  0,  5,  2,  4,  2,  0,  0,  5,  2,  0,  0,  0,  5,
       3,  2,
               1,  2,  1,  4,  4,  0,  2,  4,  4,  1,  4,  3,  1,  1,  5,
       4,  0,
               1,  0,  1,  1,  4,  1,  3,  1,  4, -1,  2,  4,  3,  4,  4,
       4,  2,
               1,  1,  4,  2,  4,  5,  1,  4,  4,  0,  1,  0,  1,  2,  4,
       1,  4,
              -1,  4,  2,  2,  4,  5,  0,  0,  4,  4,  0,  1,  0,  1,  0,
       0,  4,
               4,  0,  2,  0,  1,  0,  0,  4,  0,  0,  3,  1,  3,  4,  1,
       0,  3,
               4,  0,  4,  1,  4,  1,  0,  0,  4,  4,  0,  4,  4,  2,  2,
       5,  4,
               0,  4,  1,  4,  1,  4,  3,  4,  1,  0,  0,  2,  1,  4,  4,
       1,  3,
               1,  2,  2,  0,  3,  1,  0,  4,  0,  4,  0,  0,  4,  0,  3,
       3,  0,
```

```
       0,  4,  1,  1,  0,  2,  1,  4,  1,  0,  5,  0,  1,  4])
```

In [33]:
```
data_pca.shape
```

Out[33]:
```
(1000, 5)
```

In [34]:
```
sns.lmplot('c1', 'c2', data=data_pca, fit_reg=False, hue = 'Claster
plt.title('DBSCAN finds 2 clusters and noise')
plt.show()
```



Klasteryzacja KMeans na pliku data_dummies, bez PCA

In [35]:
```
data_dummies1 = pd.read_csv('data_dummies.csv')
data_dummies1.drop(['loan_status', 'Unnamed: 0'], axis=1, inplace=T
data_dummies1
```

Out[35]:

	loan_amnt	funded_amnt	term	int_rate	installment	sub_grade	emp_length	annual
0	5000.0	5000.0	1	10.65	162.87	1	10	240
1	2500.0	2500.0	2	15.27	59.83	2	1	300
2	2400.0	2400.0	1	15.96	84.33	3	10	122
3	10000.0	10000.0	1	13.49	339.31	4	10	492
4	3000.0	3000.0	2	12.69	67.79	5	1	800
...	...	...	...	...	...	...	...	
42530	3500.0	3500.0	1	10.28	113.39	4	1	1800
42531	1000.0	1000.0	1	9.64	32.11	14	1	120
42532	2525.0	2525.0	1	9.33	80.69	13	1	1100
42533	6500.0	6500.0	1	8.38	204.84	18	1	600
42534	5000.0	5000.0	1	7.75	156.11	17	10	700

42535 rows × 97 columns

In [36]:
```
scaler = StandardScaler()
scaler.fit(data_dummies1)
data_dummies1_std = scaler.transform(data_dummies1)
```

In [37]:
```
kmeans = KMeans(n_clusters=5, max_iter=1000)
kmeans.fit(data_dummies1)
```

Out[37]:
```
KMeans(max_iter=1000, n_clusters=5)
```

In [38]:
```
kmeans.labels_
```

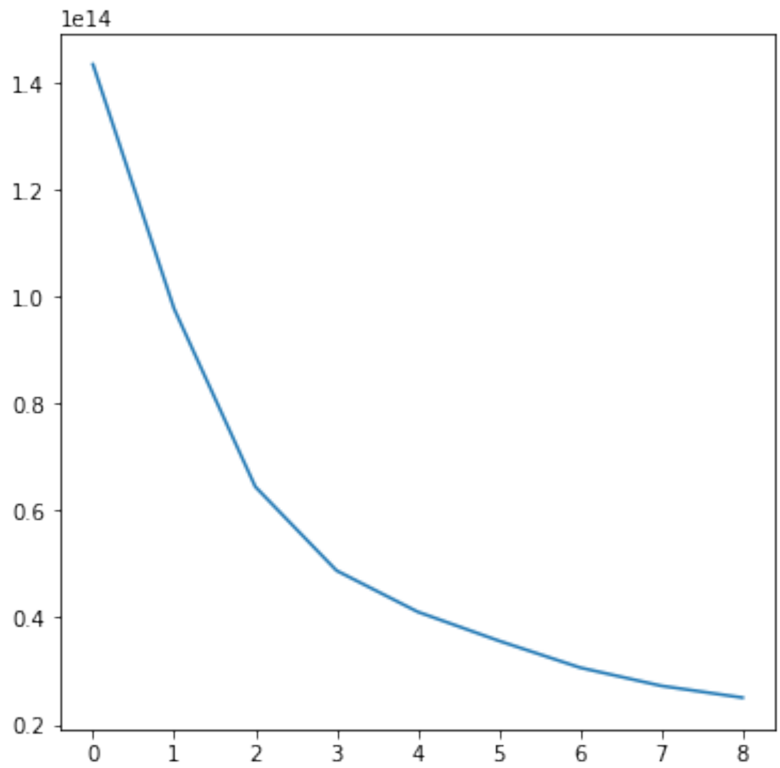
Out[38]:
```
array([2, 2, 2, ..., 0, 2, 2], dtype=int32)
```

Sprawdzamy dwie metody, ile klastrów powinniśmy zastosować

Metoda łokcia

In [39]:
```
plt.figure(figsize = (6, 6))
ssd = []
range_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10]
for num_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=num_clusters, max_iter=1000)
    kmeans.fit(data_dummies1)
    
    ssd.append(kmeans.inertia_)
    
plt.plot(ssd)
```

Out[39]:
```
[<matplotlib.lines.Line2D at 0x7f9235a80fa0>]
```



Miara wewnętrzna - wskaźnik sylwetkowy

In [40]:
```
range_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10]

for num_clusters in range_n_clusters:

    kmeans = KMeans(n_clusters=num_clusters, max_iter=1000)
    kmeans.fit(data_dummies1)

    cluster_labels = kmeans.labels_

    silhouette_avg = silhouette_score(data_dummies1, cluster_labels
    print("For n_clusters={0}, the silhouette score is {1}".format(
```

```
For n_clusters=2, the silhouette score is 0.6588343799502001
For n_clusters=3, the silhouette score is 0.642437382610893
For n_clusters=4, the silhouette score is 0.5745884037476563
For n_clusters=5, the silhouette score is 0.47399062530897446
For n_clusters=6, the silhouette score is 0.40665946218163784
For n_clusters=7, the silhouette score is 0.39109244947253297
For n_clusters=8, the silhouette score is 0.38847823683736543
For n_clusters=9, the silhouette score is 0.3513368341311928
For n_clusters=10, the silhouette score is 0.35441672612612685
```

W pobliżu k=7 się praktycznie stabilizuje, zaczyna się wypłaszczać przy k=4. Przyjmuję poniżej k=4

In [41]:
```
kmeans = KMeans(n_clusters = 4, max_iter=1000, random_state=42)
kmeans.fit(data_dummies1)
```

Out[41]:
```
KMeans(max_iter=1000, n_clusters=4, random_state=42)
```

In [42]:
```
kmeans.labels_
```

Out[42]:
```
array([1, 1, 1, ..., 0, 1, 1], dtype=int32)
```

In [43]:
```
data_dummies1['K-Means_Cluster_ID'] = kmeans.labels_
```

In [44]:
```
data_dummies1.sample(20)
```

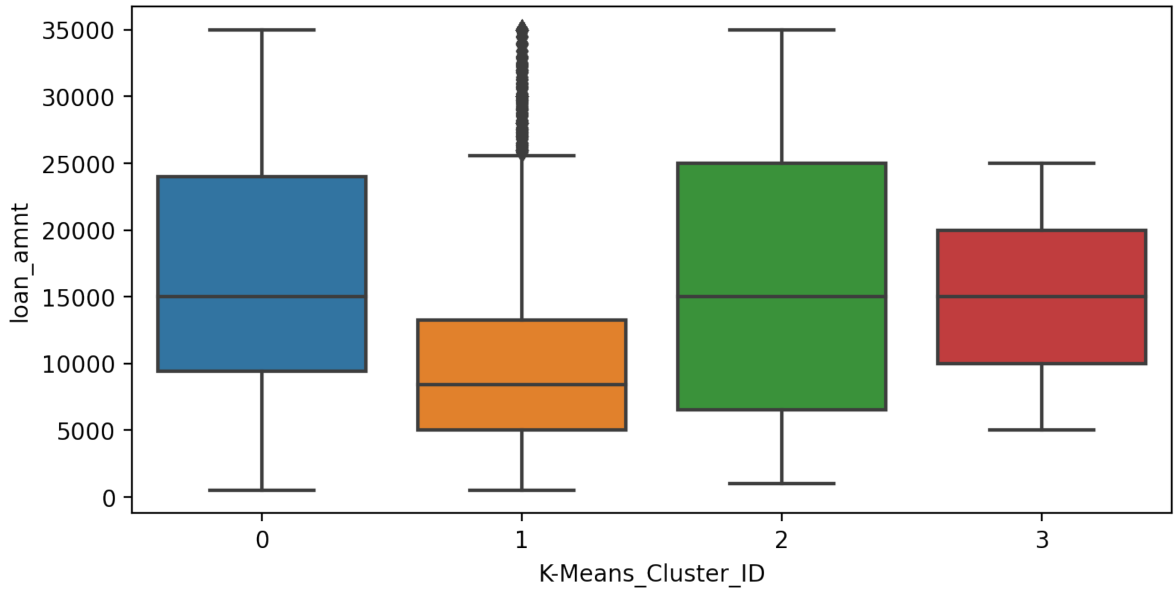
Out[44]:

	loan_amnt	funded_amnt	term	int_rate	installment	sub_grade	emp_length	annual
38146	20000.0	20000.0	1	11.78	662.19	4	7	827
2074	7750.0	7750.0	1	7.51	241.11	17	10	900
9041	4500.0	4500.0	1	5.42	135.72	12	1	670
15669	6400.0	6400.0	2	6.00	132.64	25	2	750
35612	20000.0	20000.0	1	17.04	713.49	25	9	1864
11276	3575.0	3575.0	1	11.49	117.88	14	1	500
18563	19200.0	19200.0	2	15.28	459.60	22	10	829
11954	18500.0	18500.0	1	12.99	623.25	4	10	600
6068	5500.0	5500.0	1	13.49	186.62	4	1	700
37358	5500.0	5500.0	1	7.68	171.55	20	4	700
33856	3500.0	3500.0	1	7.74	109.27	17	3	650
35494	16000.0	16000.0	1	12.87	538.14	4	1	520
41251	15000.0	15000.0	1	17.90	541.50	24	1	950
31999	10400.0	10400.0	1	10.25	336.81	1	5	450
10280	17500.0	17500.0	2	20.99	473.34	24	3	450
35978	7000.0	7000.0	1	17.58	251.60	8	2	300
32600	10000.0	10000.0	1	10.25	323.85	1	10	670
10692	8000.0	8000.0	1	5.42	241.28	12	10	620
18696	4200.0	4200.0	1	12.68	140.87	4	1	570
101	16000.0	16000.0	2	17.58	402.65	23	7	650

20 rows × 98 columns

In [45]:
```
plt.figure(figsize=(8,4),dpi=200)
sns.boxplot(x='K-Means_Cluster_ID', y='loan_amnt', data=data_dummie
```

Out[45]: <AxesSubplot:xlabel='K-Means_Cluster_ID', ylabel='loan_amnt'>

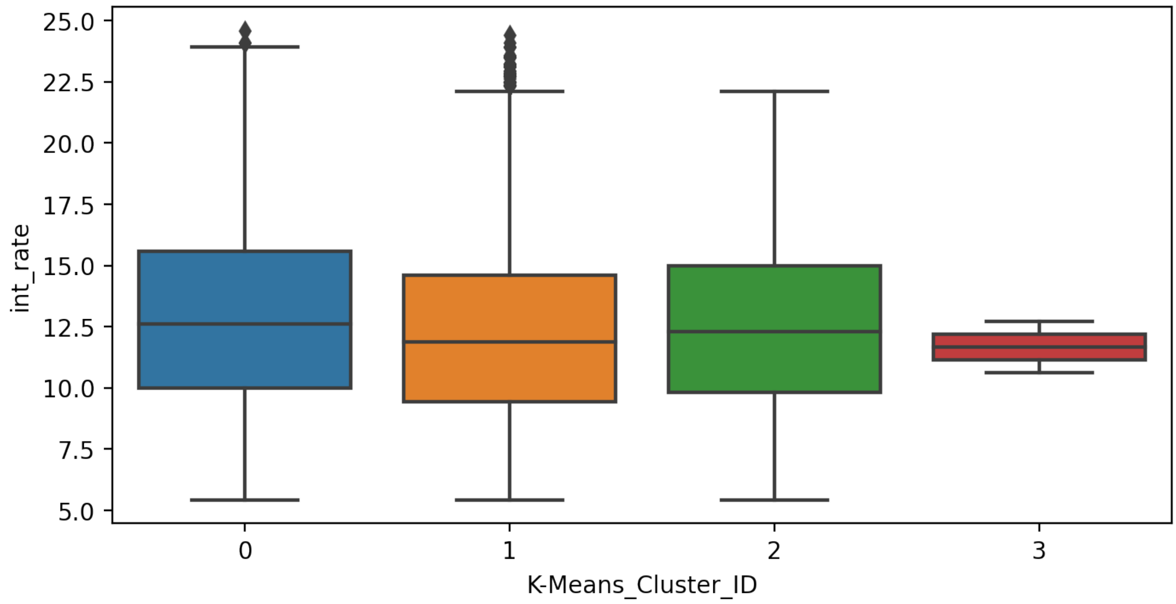


Wniosek:

W klastrze nr 2 jest najwięcej pożyczek od 7,5 tys do 25 tys. Niższe pozyczki są w klastrze 1.

In [46]:
```
plt.figure(figsize=(8,4),dpi=200)
sns.boxplot(x='K-Means_Cluster_ID', y='int_rate', data=data_dummies
```

Out[46]: <AxesSubplot:xlabel='K-Means_Cluster_ID', ylabel='int_rate'>

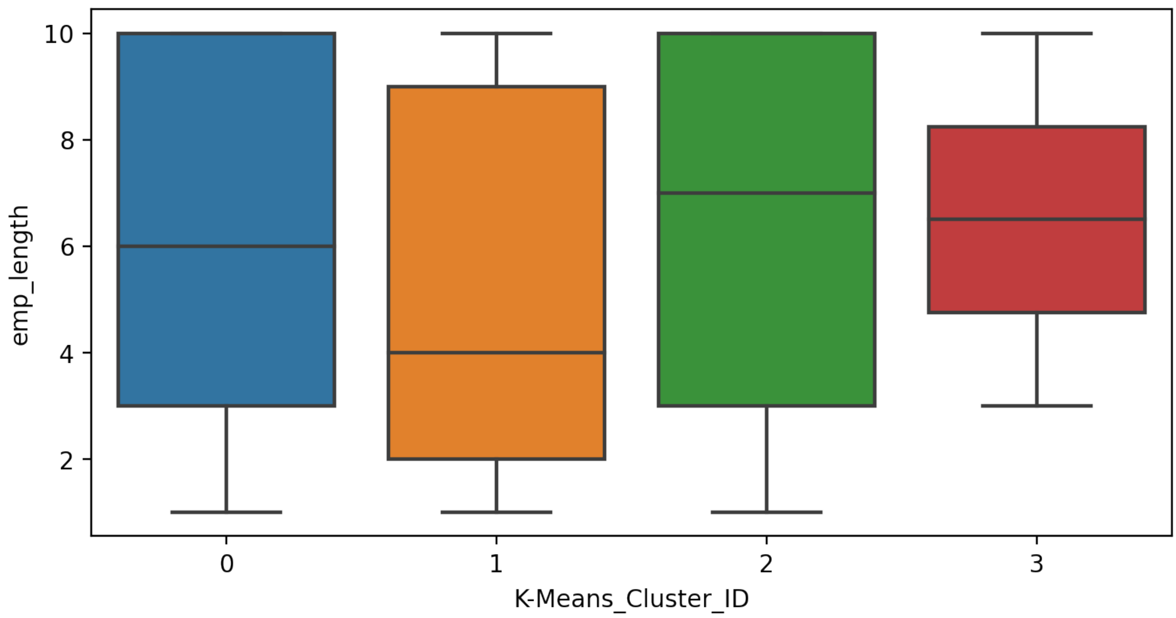


Wnioski:

W tym wypadku własciwie klastry się #zawierają w sobie" co do wysokości oprocentowania.

In [47]:
```
plt.figure(figsize=(8,4),dpi=200)
sns.boxplot(x='K-Means_Cluster_ID', y='emp_length', data=data_dummi
```

Out[47]:
```
<AxesSubplot:xlabel='K-Means_Cluster_ID', ylabel='emp_length'>
```

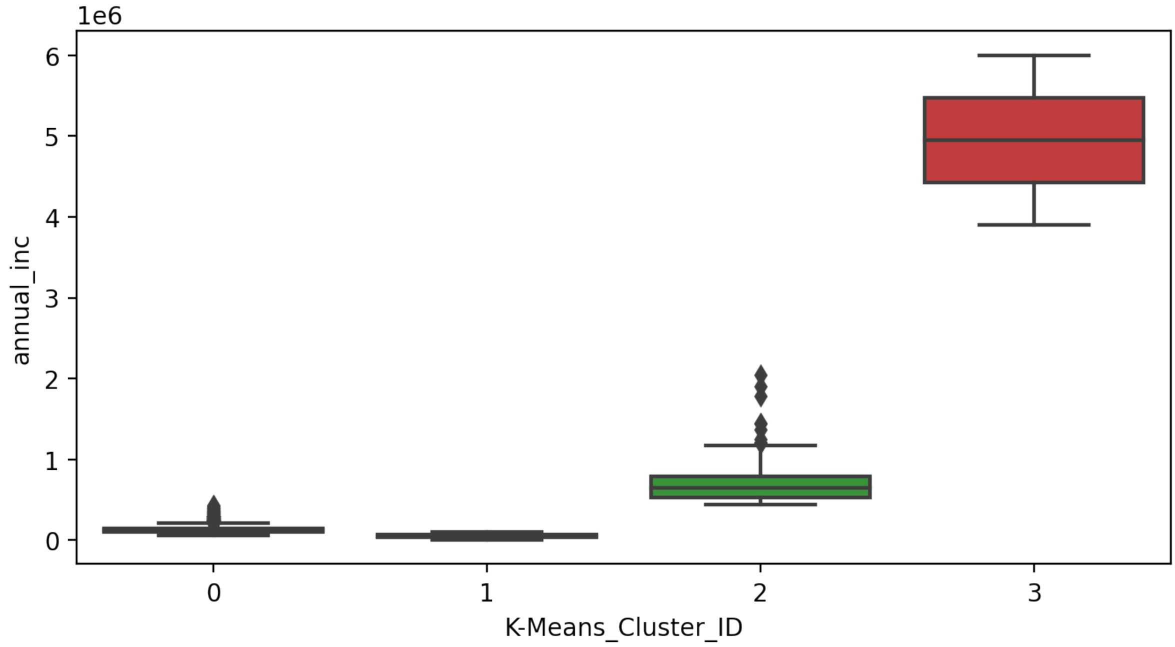


Wnioski:

Grupa pożyzckobiorców 5-9 lat została wrzucona do wszystkich klastrów, ale już ci z najniższym stażem pracy są w klastrze 1.

In [48]:
```
plt.figure(figsize=(8,4),dpi=200)
sns.boxplot(x='K-Means_Cluster_ID', y='annual_inc', data=data_dummi
```

Out[48]: <AxesSubplot:xlabel='K-Means_Cluster_ID', ylabel='annual_inc'>



Wnioski:

Pożyczkobiorcy z najwyższym dochodem zostali wrzuceni do grupy 3.

Kolejna część w pliku "Modelowanie III część Projekt końcowy Data Science - Dorota Gawrońska-Popa"

Łódź 4.10.2020