k-modes

December 25, 2023

[5]:
```
# Impor beberapa Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from kmodes.kmodes import KModes
```

[6]:
```
# import data Kuesioner
df = pd.read_csv('Kuesioner.csv')
# Menampilkan data dari 5 baris teratas
df.head()
```

[6]:

	Jenis Kelamin	Usia	Status	Anggaran Belanja Pakaian	Domisili	\
0	1	2	1	3	1	
1	1	2	2	2	5	
2	2	2	1	1	5	
3	2	2	2	4	5	
4	2	2	1	2	10	

	Frekuensi Pembelian	TRU1	TRU2	TRU3	TRU4	…	FK	FM	FCS	FTH	TMF	\
0	3	4	5	4	4	…	4	5	2	5	4	
1	6	4	5	5	4	…	5	5	5	5	5	
2	7	4	3	3	4	…	5	3	3	5	3	
3	3	4	5	4	4	…	4	5	5	4	4	
4	7	5	5	5	5	…	5	5	5	5	5	

	FD	REL1	REL2	REL3	REL4
0	2	4	5	2	4
1	5	5	2	5	5
2	5	4	4	5	5
3	5	4	4	4	5
4	5	5	5	5	5

[5 rows x 23 columns]

[7]:
```
df.info()
```

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

```
RangeIndex: 150 entries, 0 to 149
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Jenis Kelamin             150 non-null    int64
 1   Usia                      150 non-null    int64
 2   Status                    150 non-null    int64
 3   Anggaran Belanja Pakaian  150 non-null    int64
 4   Domisili                  150 non-null    int64
 5   Frekuensi Pembelian       150 non-null    int64
 6   TRU1                      150 non-null    int64
 7   TRU2                      150 non-null    int64
 8   TRU3                      150 non-null    int64
 9   TRU4                      150 non-null    int64
 10  FB                        150 non-null    int64
 11  FI                        150 non-null    int64
 12  FH                        150 non-null    int64
 13  FK                        150 non-null    int64
 14  FM                        150 non-null    int64
 15  FCS                       150 non-null    int64
 16  FTH                       150 non-null    int64
 17  TMF                       150 non-null    int64
 18  FD                        150 non-null    int64
 19  REL1                      150 non-null    int64
 20  REL2                      150 non-null    int64
 21  REL3                      150 non-null    int64
 22  REL4                      150 non-null    int64
dtypes: int64(23)
memory usage: 27.1 KB
```

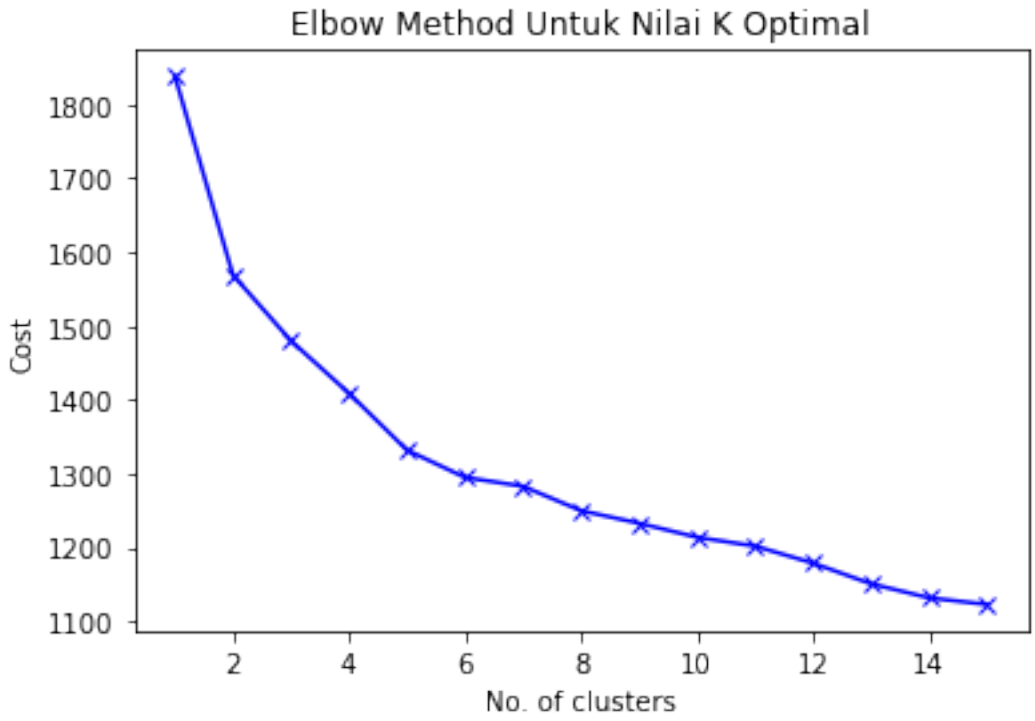
[46]:
```
# Elbow Methods untuk menentukan nilai k yang optimal
cost = []
K = range(1,16)
for num_clusters in list(K):
    kmode = KModes(n_clusters=num_clusters, init = "Cao", n_init = 15,␣
 ↪verbose=1)
    kmode.fit_predict(df)
    cost.append(kmode.cost_)

plt.plot(K, cost, 'bx-')
plt.xlabel('No. of clusters')
plt.ylabel('Cost')
plt.title('Elbow Method Untuk Nilai K Optimal')
plt.show()
```

```
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
```

```
Starting iterations…
Run 1, iteration: 1/100, moves: 0, cost: 1838.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 26, cost: 1613.0
Run 1, iteration: 2/100, moves: 27, cost: 1573.0
Run 1, iteration: 3/100, moves: 7, cost: 1569.0
Run 1, iteration: 4/100, moves: 1, cost: 1569.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 40, cost: 1490.0
Run 1, iteration: 2/100, moves: 12, cost: 1482.0
Run 1, iteration: 3/100, moves: 6, cost: 1480.0
Run 1, iteration: 4/100, moves: 1, cost: 1480.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 43, cost: 1438.0
Run 1, iteration: 2/100, moves: 23, cost: 1410.0
Run 1, iteration: 3/100, moves: 3, cost: 1409.0
Run 1, iteration: 4/100, moves: 0, cost: 1409.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 47, cost: 1356.0
Run 1, iteration: 2/100, moves: 18, cost: 1334.0
Run 1, iteration: 3/100, moves: 6, cost: 1332.0
Run 1, iteration: 4/100, moves: 2, cost: 1332.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 40, cost: 1346.0
Run 1, iteration: 2/100, moves: 20, cost: 1308.0
Run 1, iteration: 3/100, moves: 15, cost: 1295.0
Run 1, iteration: 4/100, moves: 4, cost: 1295.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 46, cost: 1301.0
Run 1, iteration: 2/100, moves: 12, cost: 1294.0
```

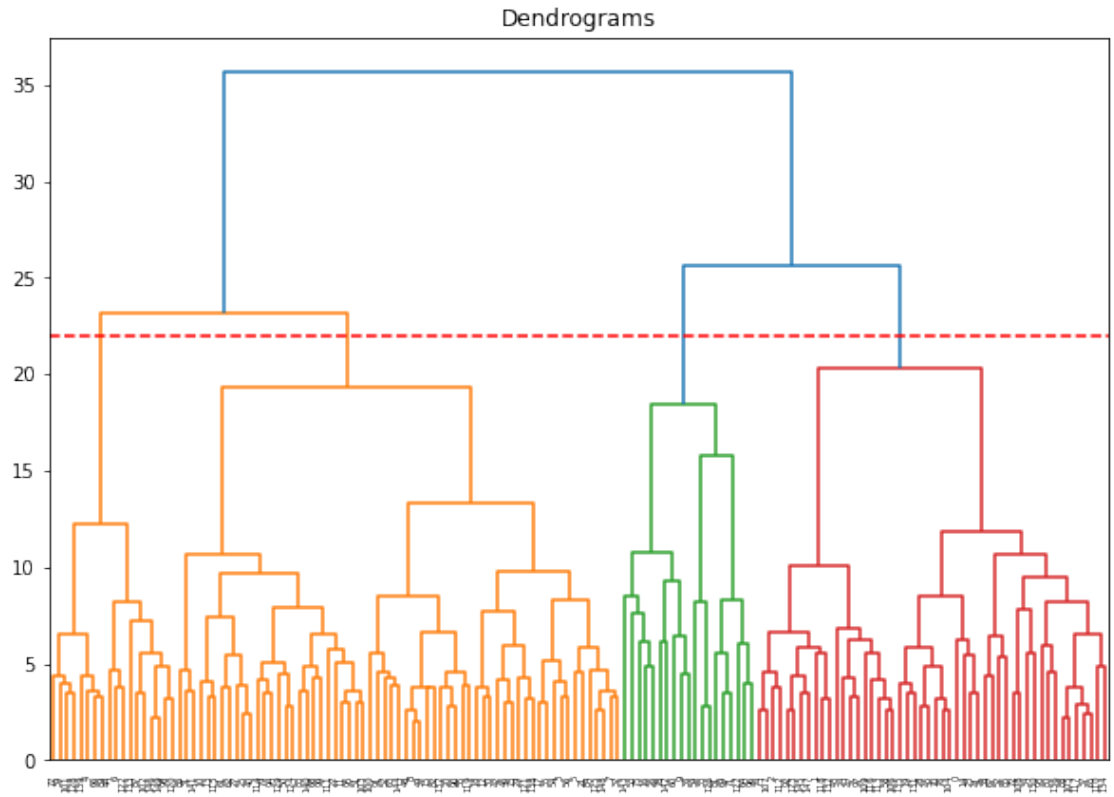
```
Run 1, iteration: 3/100, moves: 5, cost: 1283.0
Run 1, iteration: 4/100, moves: 2, cost: 1283.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 52, cost: 1271.0
Run 1, iteration: 2/100, moves: 15, cost: 1261.0
Run 1, iteration: 3/100, moves: 5, cost: 1250.0
Run 1, iteration: 4/100, moves: 2, cost: 1250.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 41, cost: 1255.0
Run 1, iteration: 2/100, moves: 17, cost: 1241.0
Run 1, iteration: 3/100, moves: 3, cost: 1236.0
Run 1, iteration: 4/100, moves: 4, cost: 1233.0
Run 1, iteration: 5/100, moves: 0, cost: 1233.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 30, cost: 1225.0
Run 1, iteration: 2/100, moves: 16, cost: 1214.0
Run 1, iteration: 3/100, moves: 0, cost: 1214.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 30, cost: 1214.0
Run 1, iteration: 2/100, moves: 16, cost: 1202.0
Run 1, iteration: 3/100, moves: 0, cost: 1202.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 36, cost: 1190.0
Run 1, iteration: 2/100, moves: 8, cost: 1184.0
Run 1, iteration: 3/100, moves: 6, cost: 1179.0
Run 1, iteration: 4/100, moves: 0, cost: 1179.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 37, cost: 1159.0
Run 1, iteration: 2/100, moves: 11, cost: 1152.0
Run 1, iteration: 3/100, moves: 2, cost: 1151.0
```

```
Run 1, iteration: 4/100, moves: 0, cost: 1151.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 32, cost: 1154.0
Run 1, iteration: 2/100, moves: 18, cost: 1137.0
Run 1, iteration: 3/100, moves: 6, cost: 1132.0
Run 1, iteration: 4/100, moves: 0, cost: 1132.0
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 28, cost: 1151.0
Run 1, iteration: 2/100, moves: 18, cost: 1129.0
Run 1, iteration: 3/100, moves: 7, cost: 1123.0
Run 1, iteration: 4/100, moves: 0, cost: 1123.0
```



[51]:
```
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(df, method='ward'))
plt.axhline(y=22, color='r', linestyle='--')
```

[51]: <matplotlib.lines.Line2D at 0x232a3c92940>



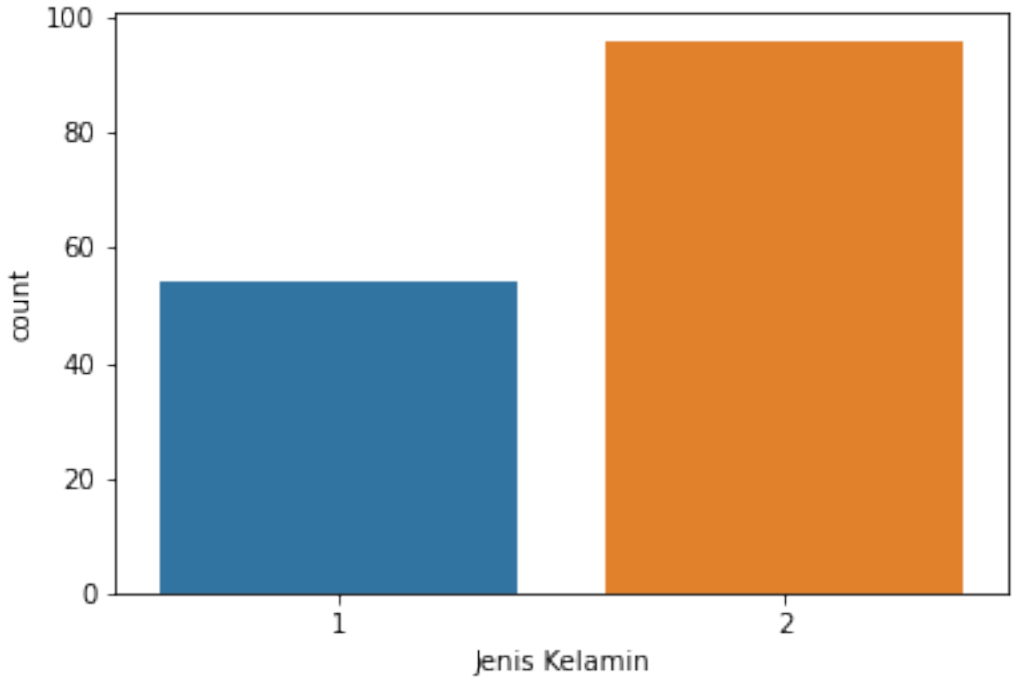
[39]:
```
kmode = KModes(n_clusters=3, init = "Cao", n_init = 15, verbose=1)
clusters = kmode.fit_predict(df)
clusters
```

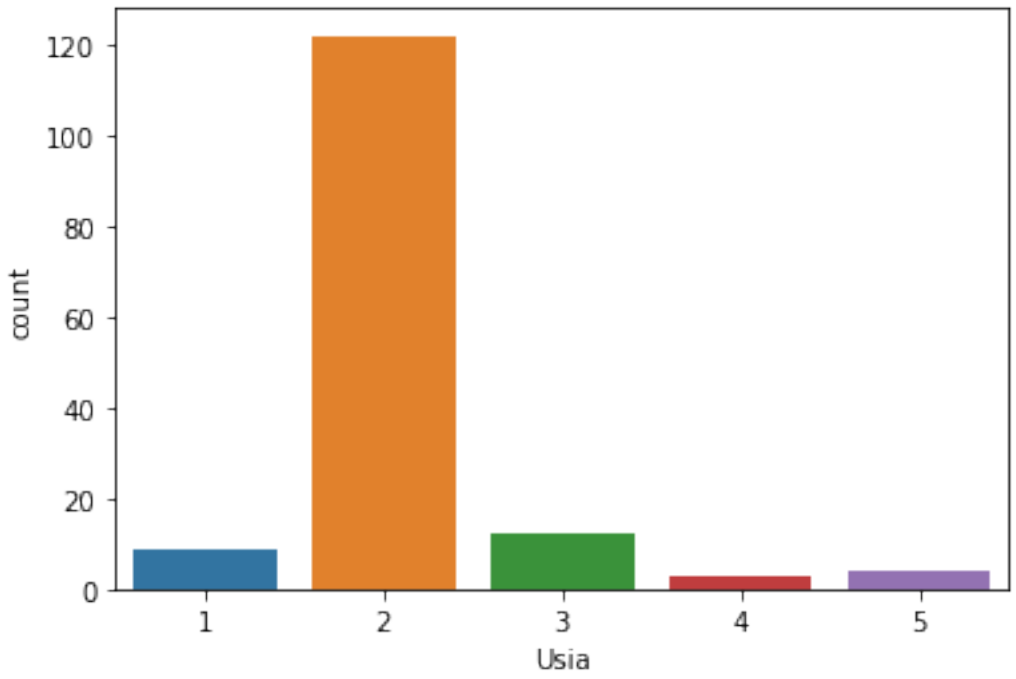
```
Initialization method and algorithm are deterministic. Setting n_init to 1.
Init: initializing centroids
Init: initializing clusters
Starting iterations…
Run 1, iteration: 1/100, moves: 40, cost: 1490.0
Run 1, iteration: 2/100, moves: 12, cost: 1482.0
Run 1, iteration: 3/100, moves: 6, cost: 1480.0
Run 1, iteration: 4/100, moves: 1, cost: 1480.0
```

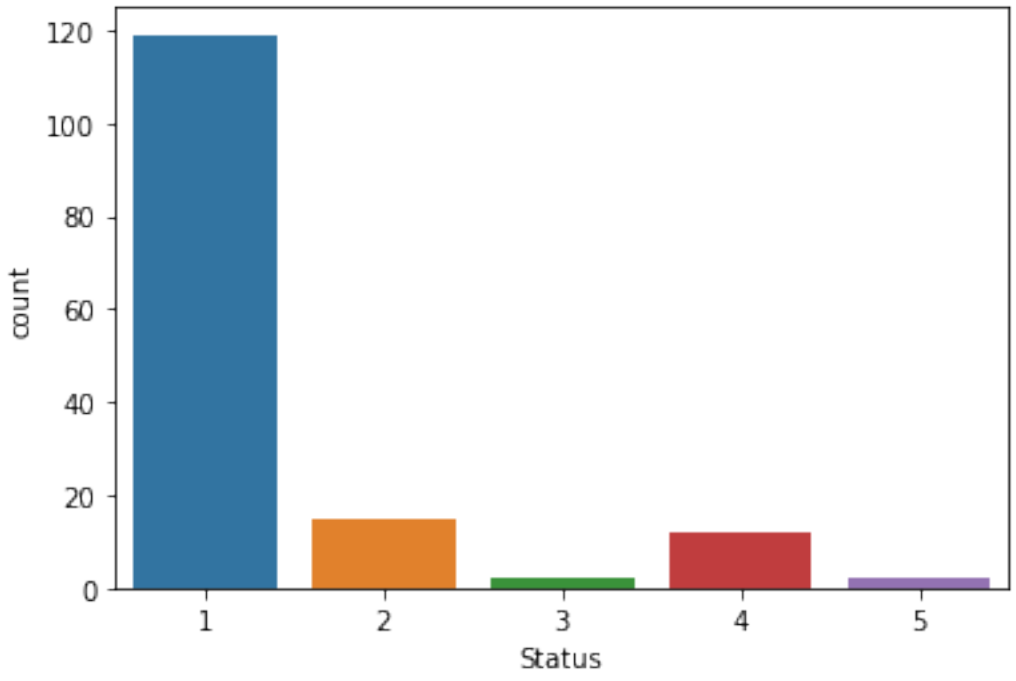
[39]:
```
array([2, 0, 0, 2, 0, 2, 0, 2, 0, 0, 2, 2, 0, 2, 2, 0, 2, 2, 0, 0, 0, 1,
       2, 2, 0, 2, 0, 0, 2, 2, 0, 1, 0, 0, 1, 2, 1, 2, 2, 0, 2, 2, 0, 0,
       2, 2, 0, 0, 0, 0, 2, 2, 1, 0, 0, 2, 2, 2, 0, 0, 1, 1, 1, 0, 1, 2,
       0, 1, 1, 1, 2, 1, 2, 0, 0, 0, 0, 0, 1, 0, 0, 1, 2, 2, 0, 0, 1, 0,
       2, 0, 0, 2, 0, 0, 0, 2, 1, 0, 0, 2, 0, 0, 1, 2, 0, 2, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 2, 0, 2, 0, 0, 1, 0, 0, 2, 0, 1, 0, 0, 2, 0,
       2, 2, 0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 2, 1, 2, 2, 1], dtype=uint16)
```

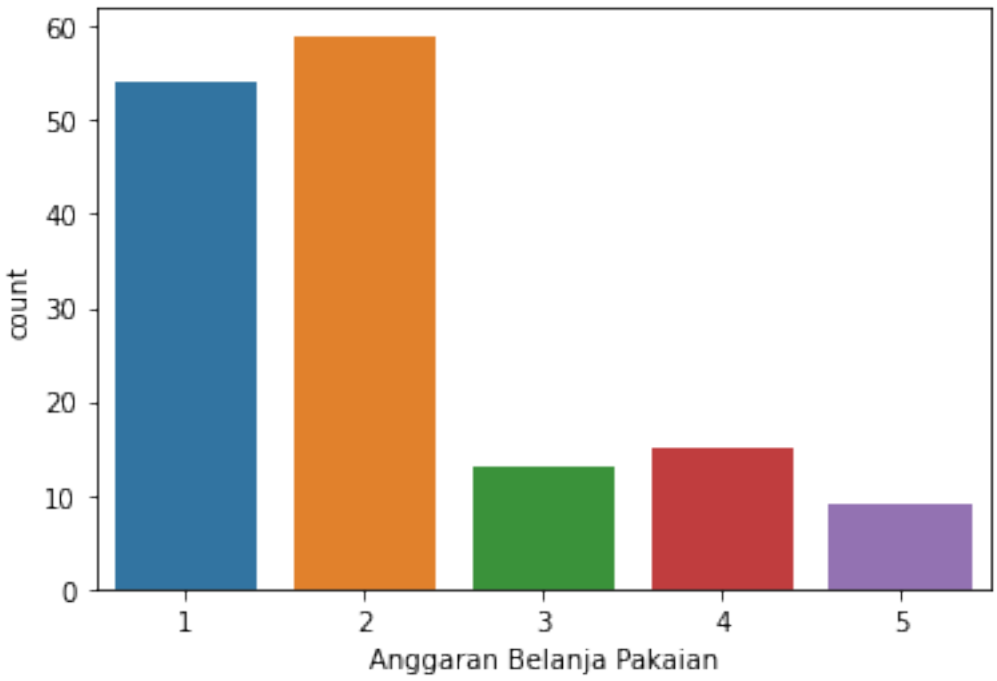
[21]:
```
# Menampilkan visualisasi data kategori dari setiap kolom
for i in df:
  plt.figure()
  sns.countplot(data=df, x=i)
```

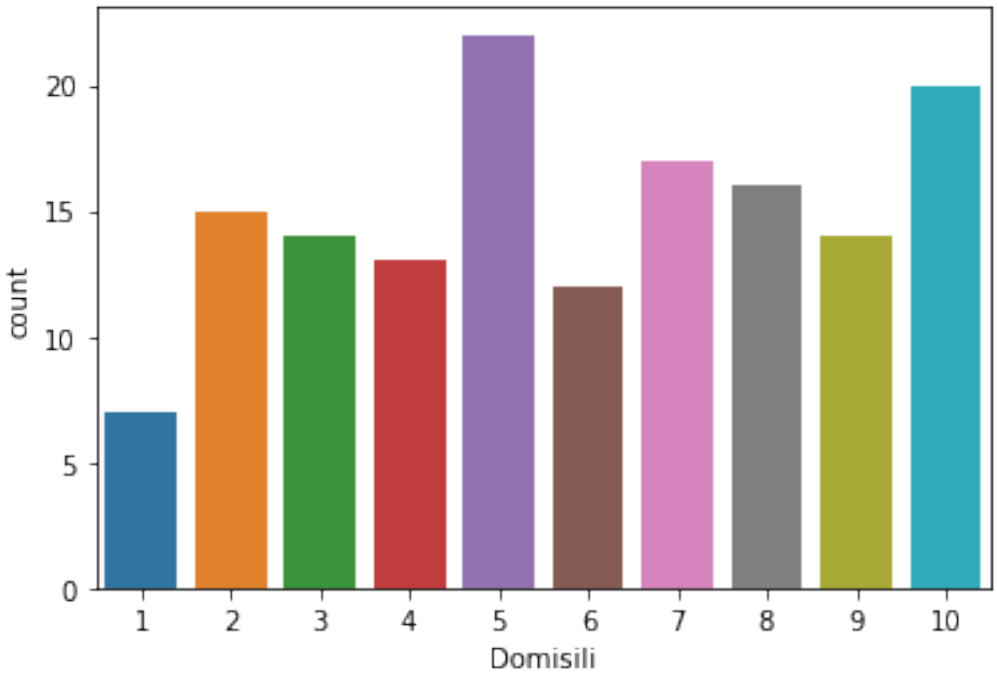
```
<ipython-input-21-f0f438a6188c>:3: RuntimeWarning: More than 20 figures have
been opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and may
consume too much memory. (To control this warning, see the rcParam
`figure.max_open_warning`).
  plt.figure()
```

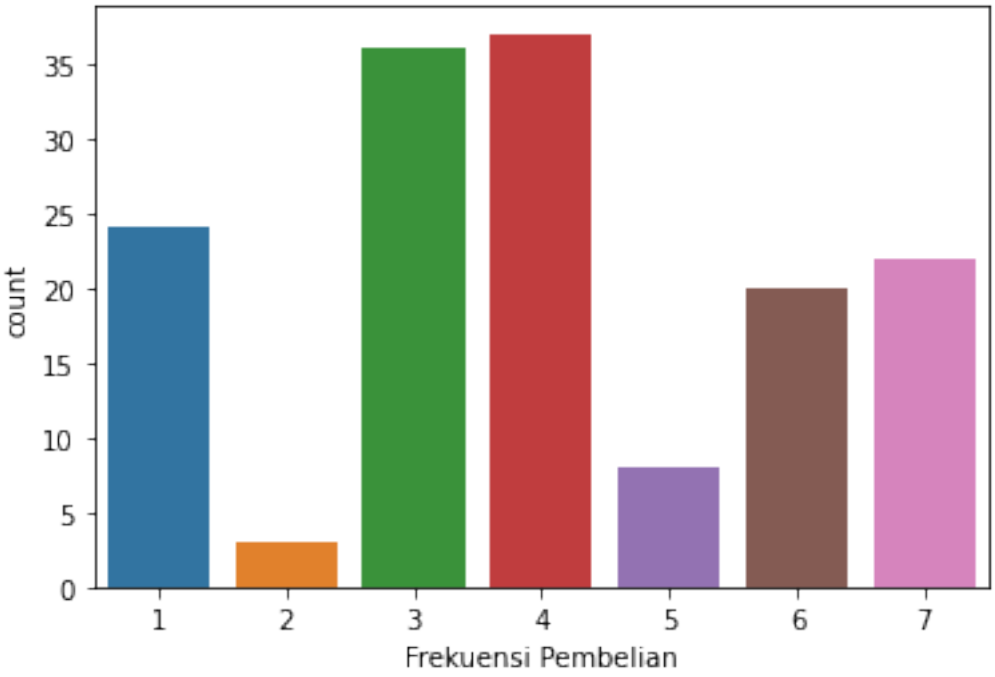


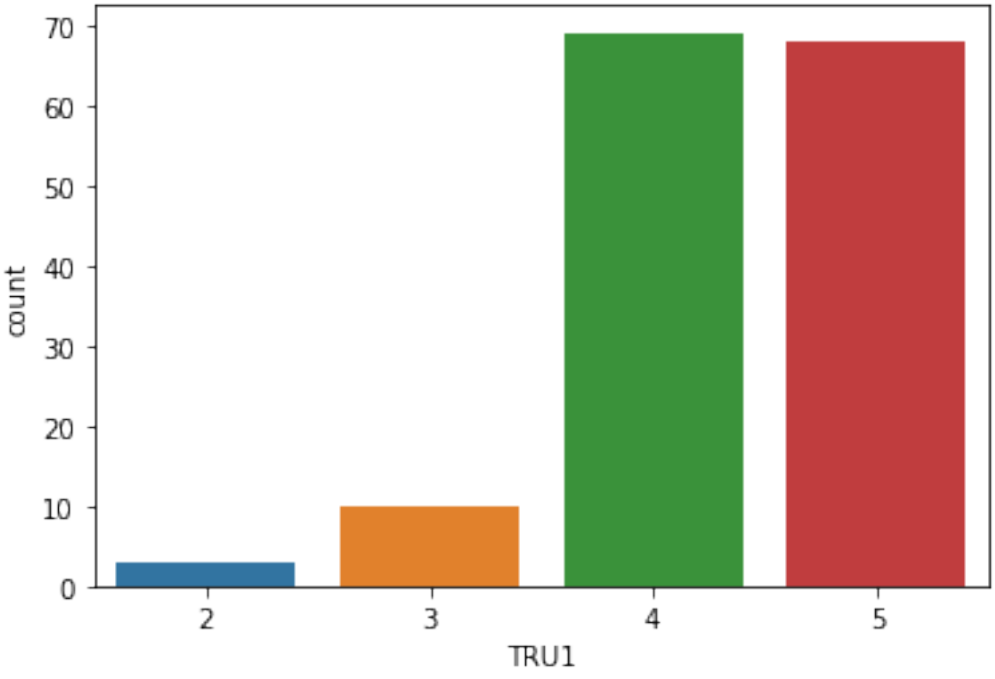

120
100
80
60
40
20
0
count
1
2
3
4
5
Usia

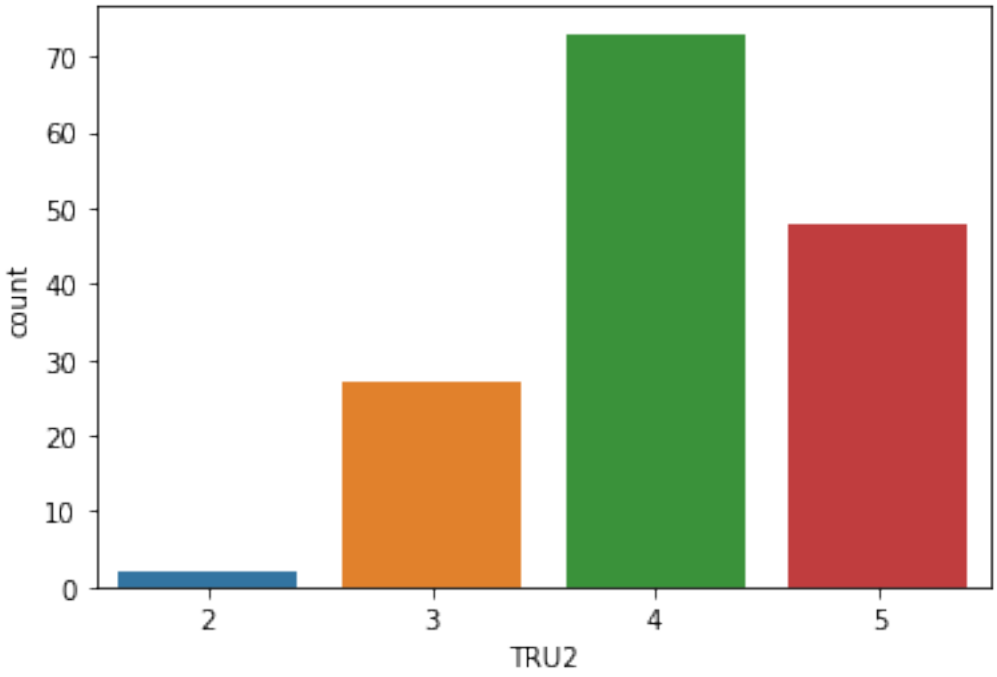

120
100
80
60
40
20
0
count
1
2
3
4
5
Status

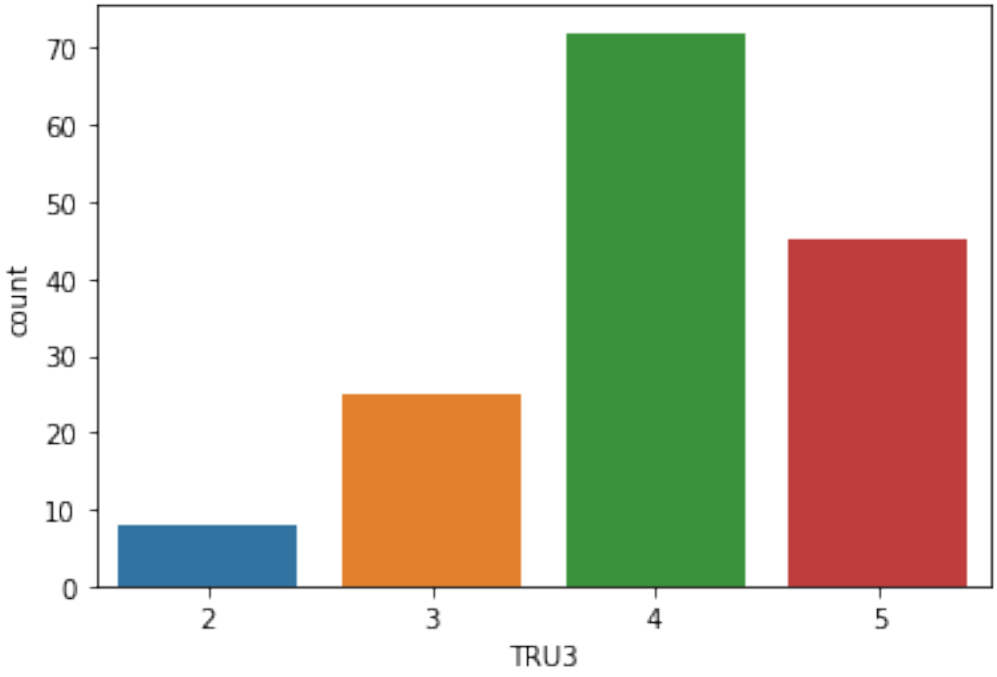

60
50
40
30
20
10
0
count
1
2
3
4
5
Anggaran Belanja Pakaian

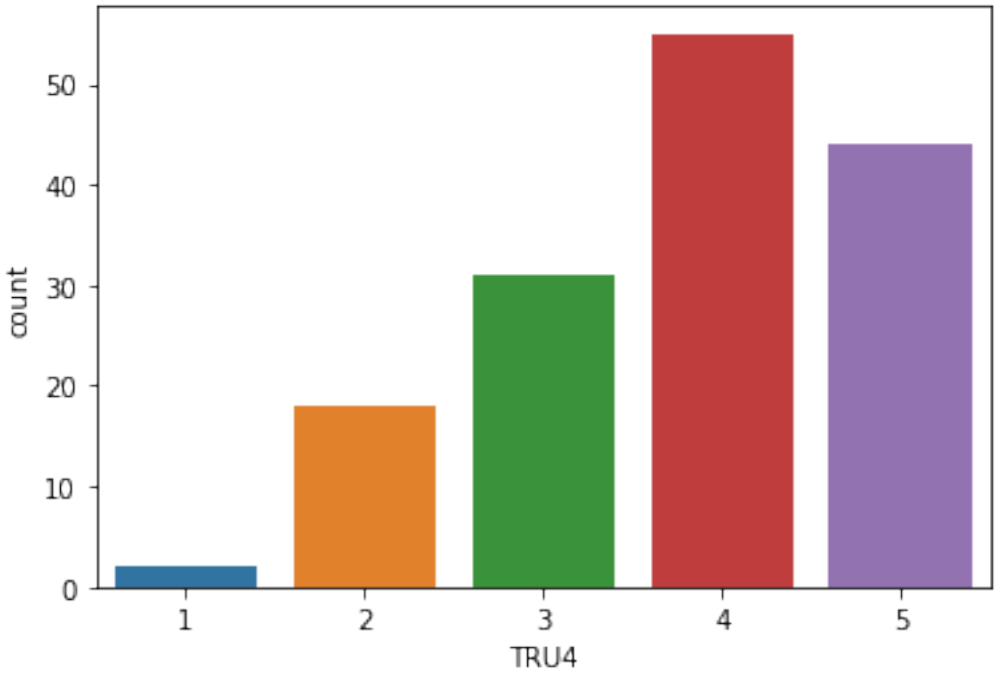

20
15
10
5
0
count
1
2
3
4
5
6
7
8
9
10
Domisili

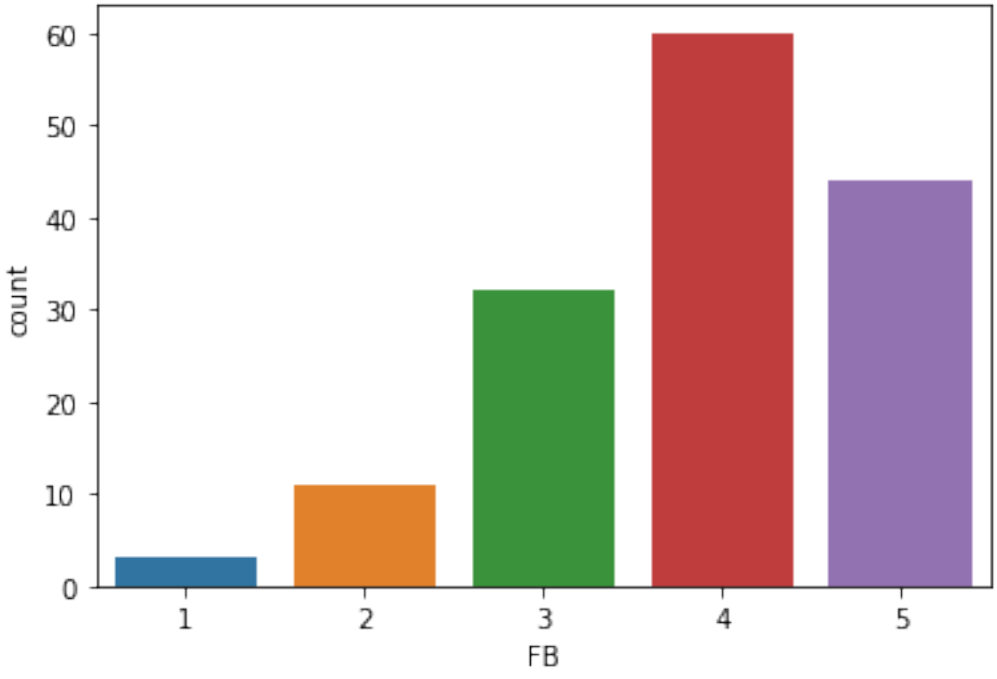

count
35
30
25
20
15
10
5
0
1
2
3
4
5
6
7
Frekuensi Pembelian

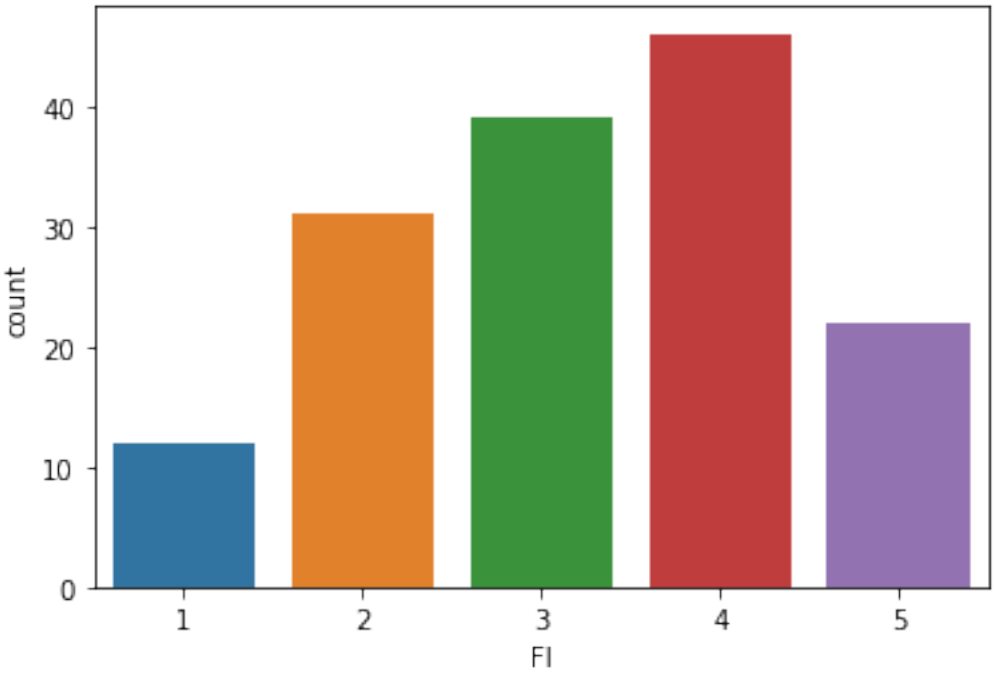

count
70
60
50
40
30
20
10
0
2
3
4
5
TRU1

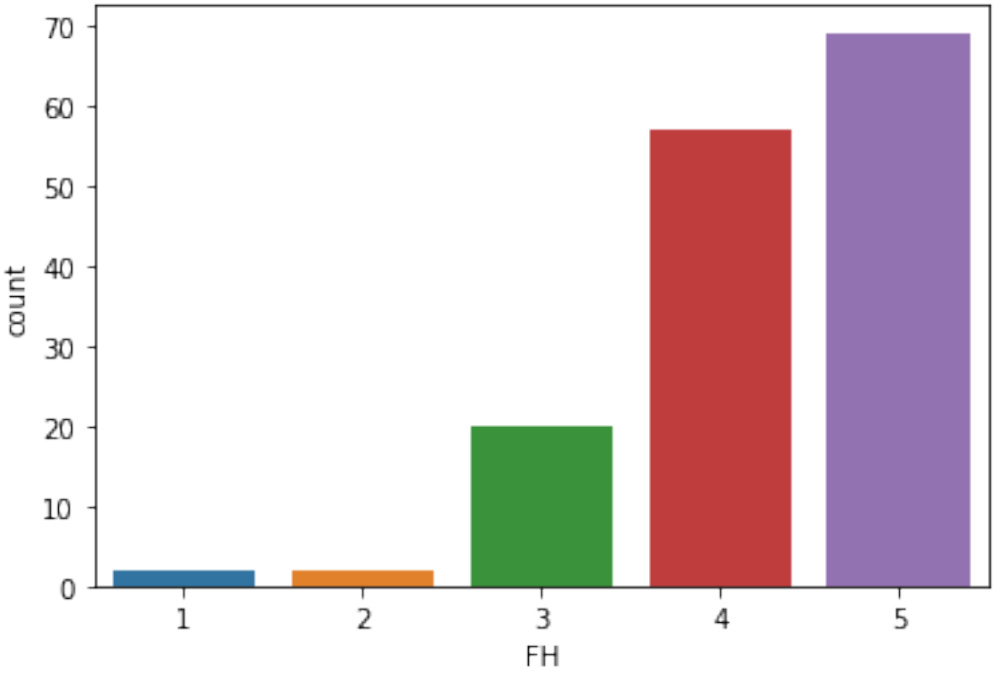

70
60
50
40
30
20
10
0
count
2
3
4
5
TRU2

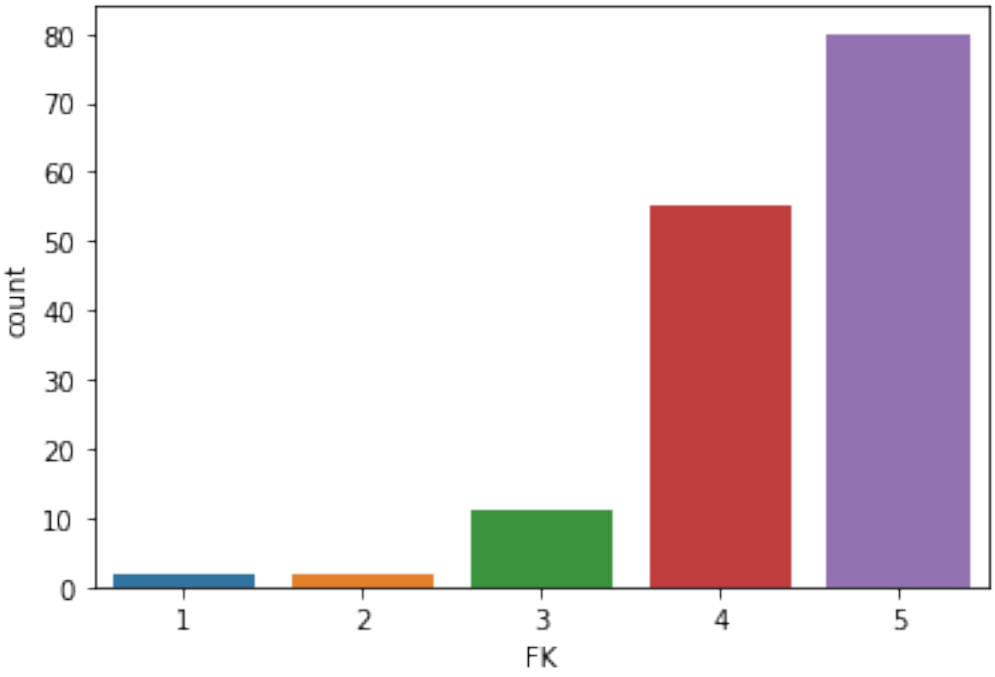

70
60
50
40
30
20
10
0
count
2
3
4
5
TRU3

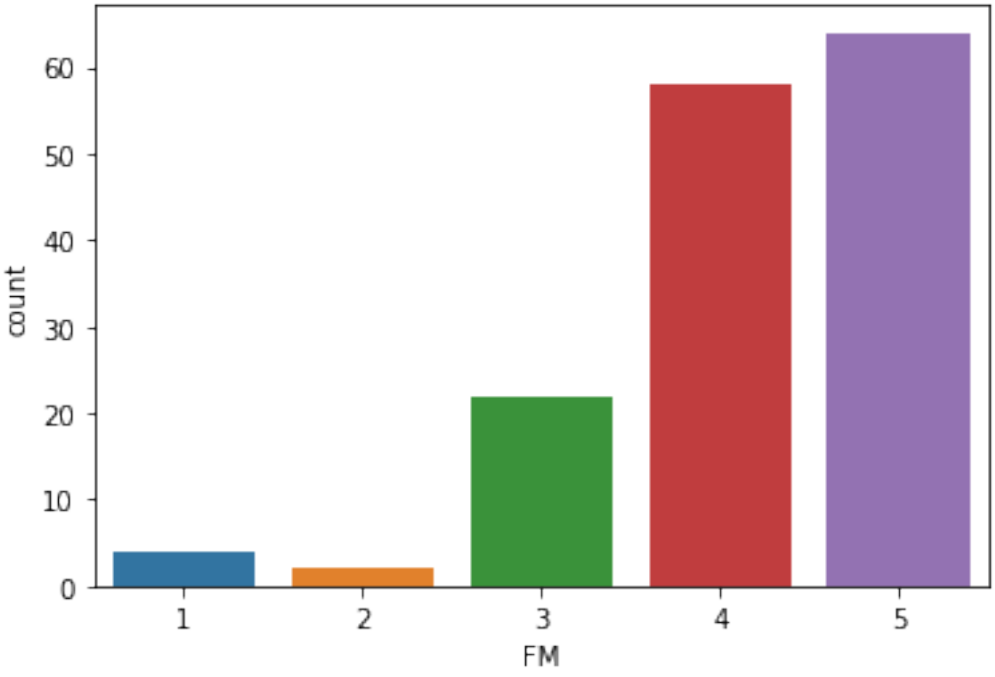

count
0
10
20
30
40
50
1
2
3
4
5
TRU4

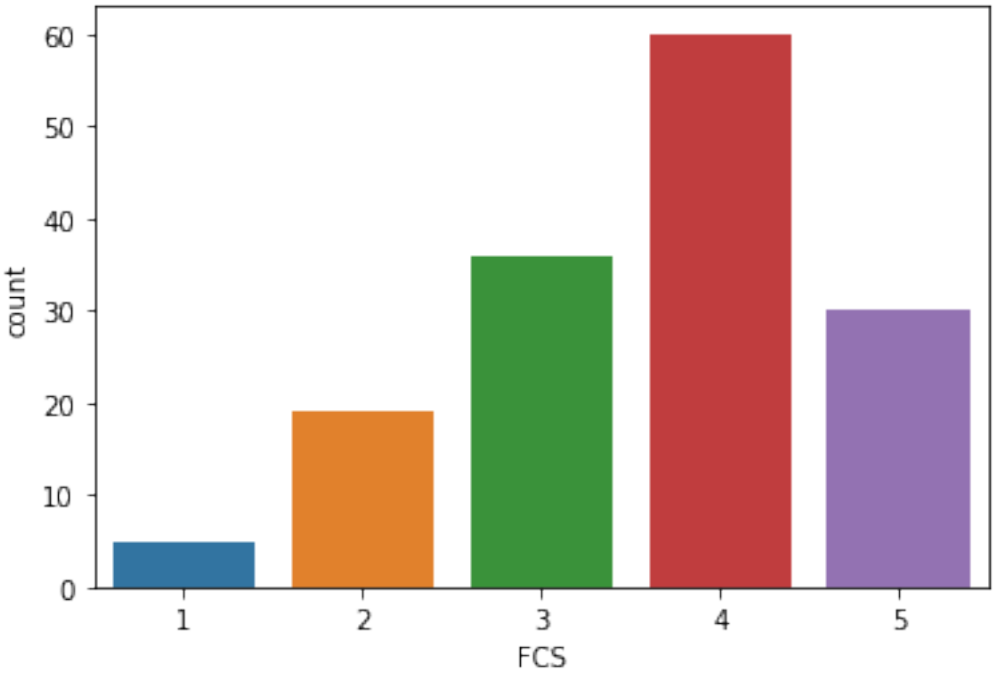

count
0
10
20
30
40
50
60
1
2
3
4
5
FB

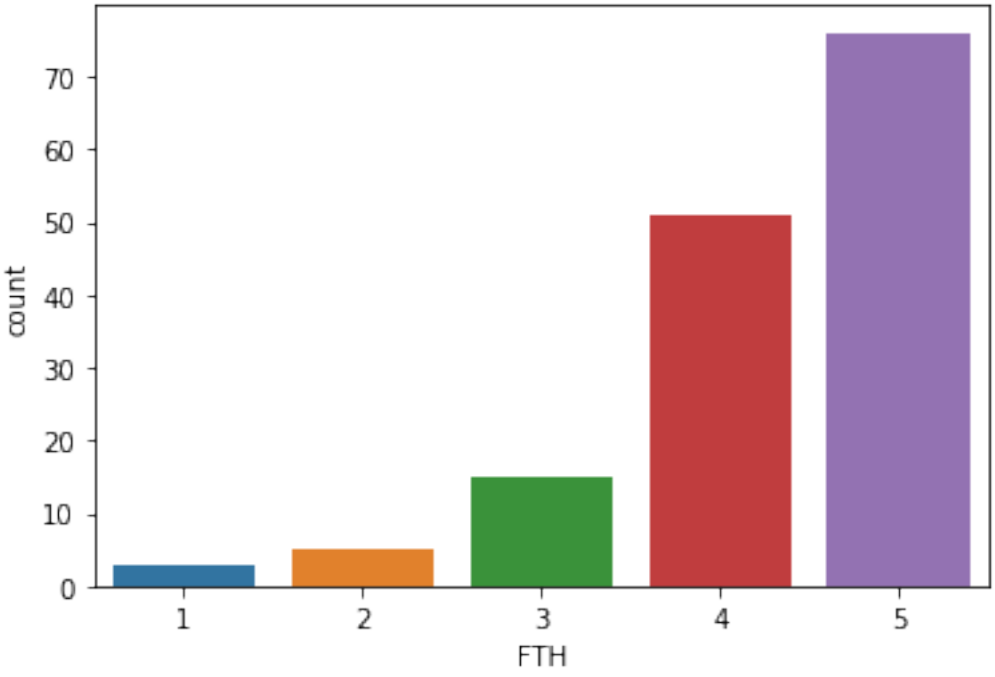

count
0
10
20
30
40
1
2
3
4
5
FI

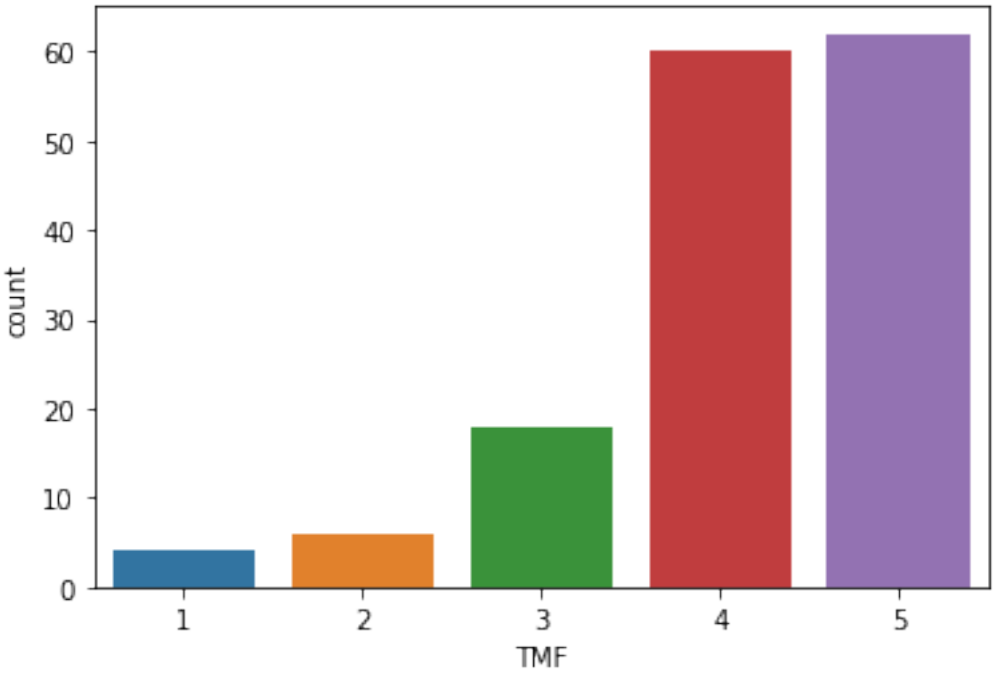

count
0
10
20
30
40
50
60
70
1
2
3
4
5
FH

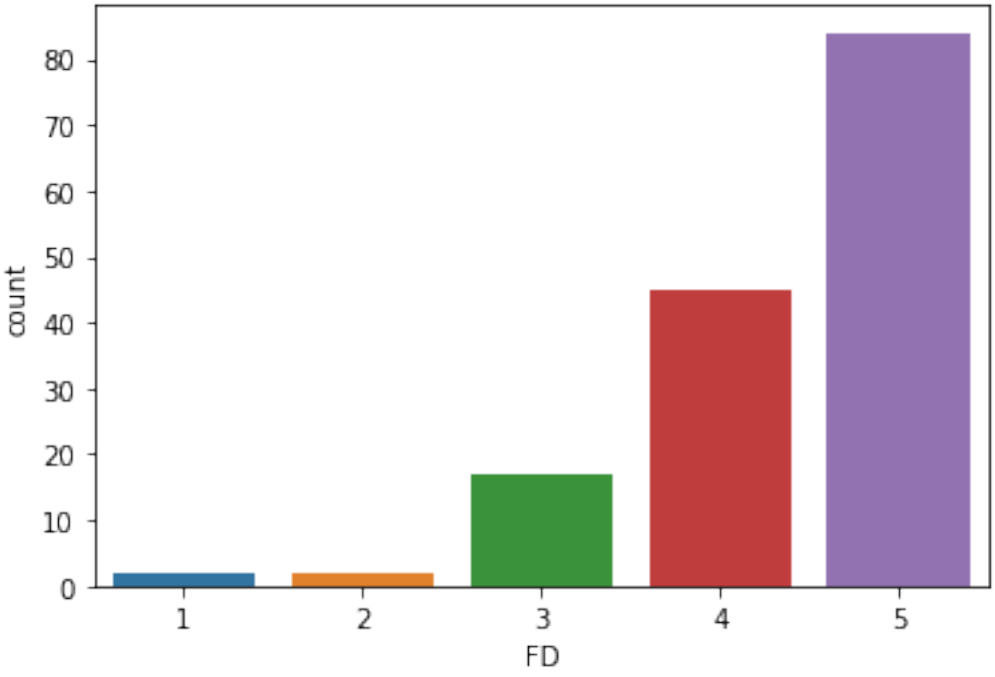

80
70
60
50
40
30
20
10
0
count
1
2
3
4
5
FK

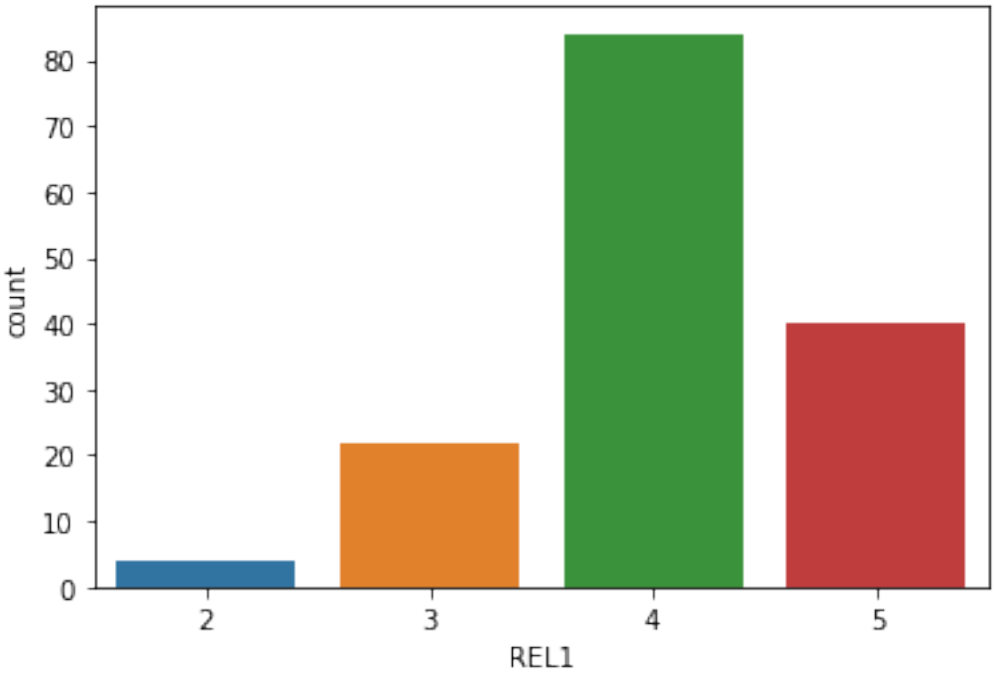

60
50
40
30
20
10
0
count
1
2
3
4
5
FM

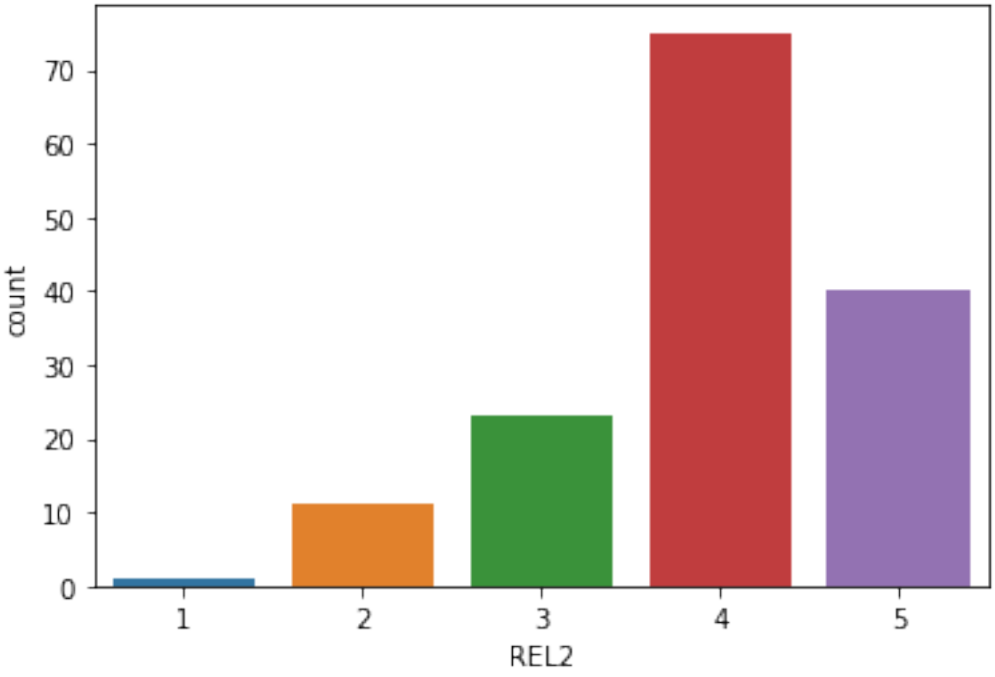

60
50
40
30
20
10
0
count
1
2
3
4
5
FCS

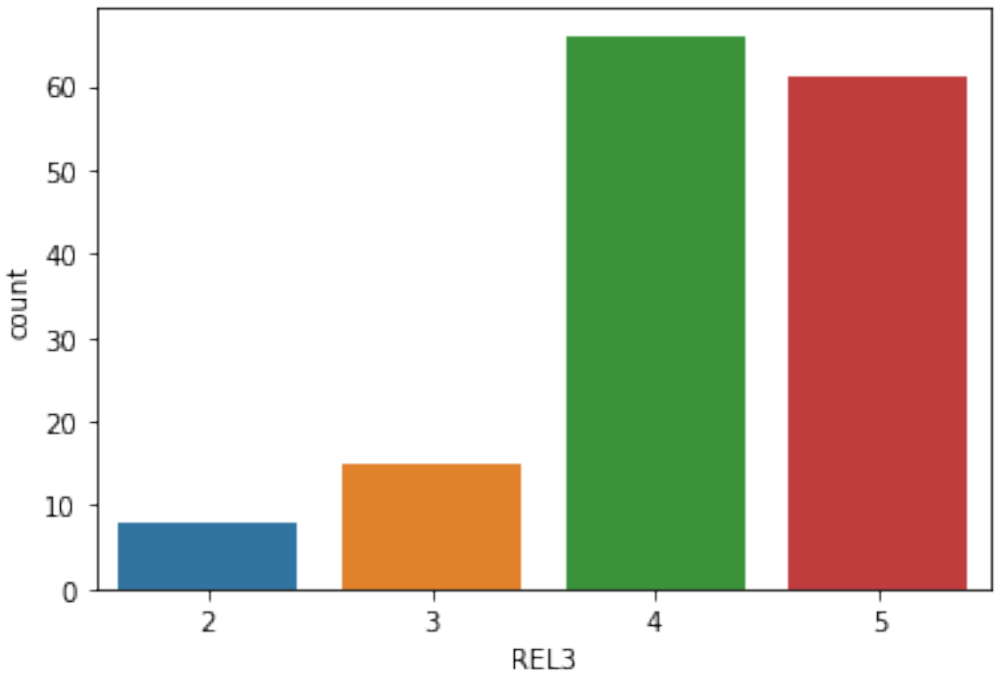

70
60
50
40
30
20
10
0
count
1
2
3
4
5
FTH

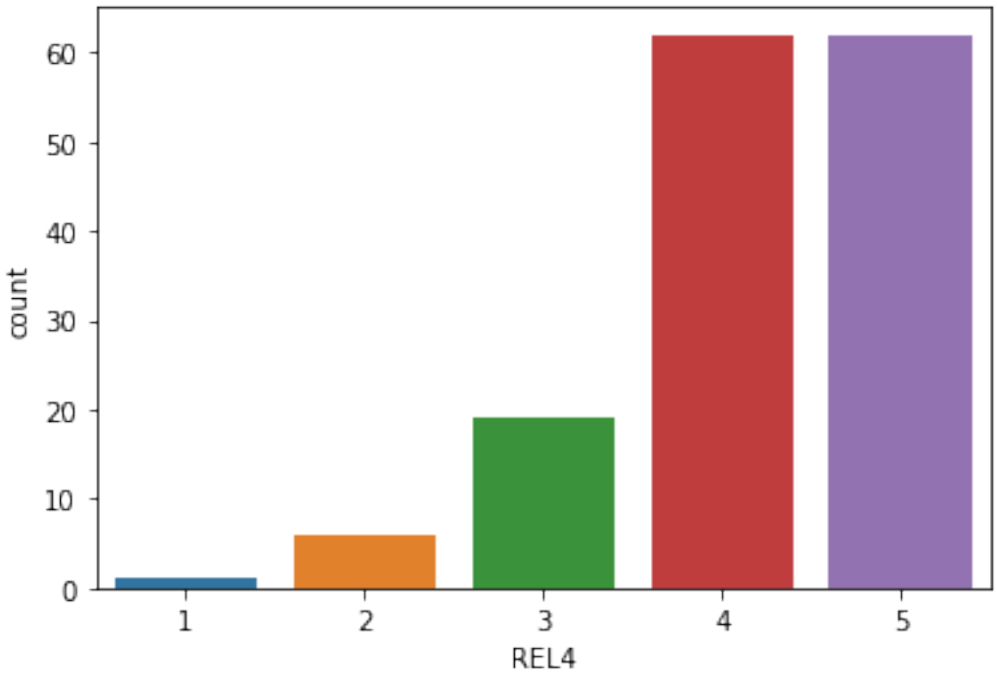

count
60
50
40
30
20
10
0
1
2
3
4
5
TMF


count
80
70
60
50
40
30
20
10
0
1
2
3
4
5
FD


count
80
70
60
50
40
30
20
10
0
2
3
4
5
REL1


count
70
60
50
40
30
20
10
0
1
2
3
4
5
REL2





[41]:
```
# Menyimpan data cluster ke variabel df yang baru
df_cluster = df.copy()
```

```
df_cluster['Cluster'] = clusters
df_cluster.head()
# Menampilkan data
df_cluster.head()
```

[41]:

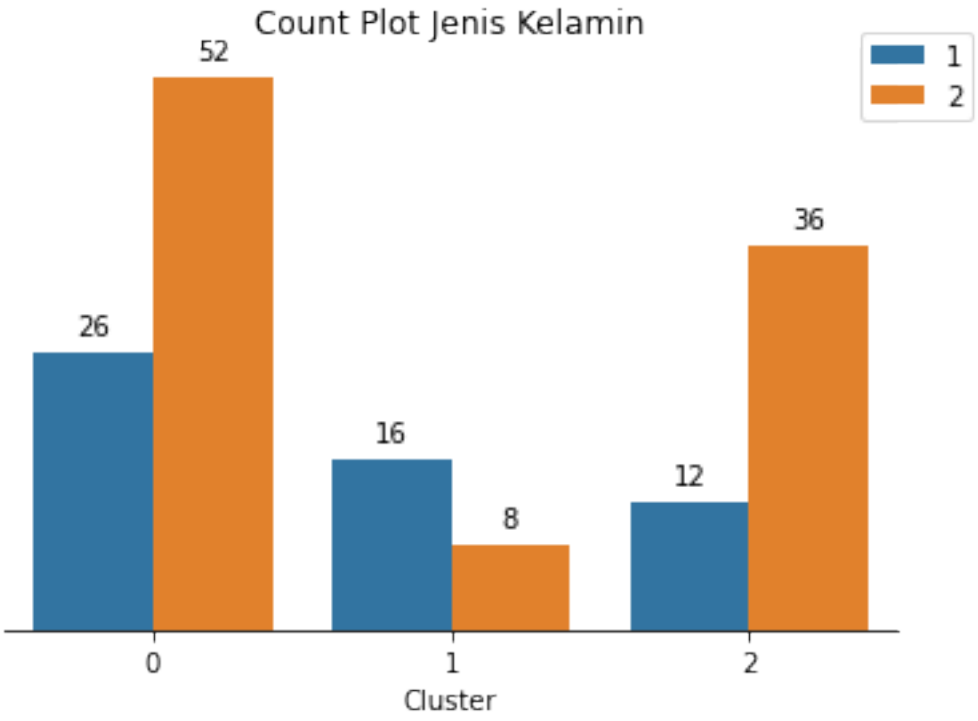
	Jenis Kelamin	Usia	Status	Anggaran Belanja Pakaian	Domisili	\
0	1	2	1	3	1	
1	1	2	2	2	5	
2	2	2	1	1	5	
3	2	2	2	4	5	
4	2	2	1	2	10	

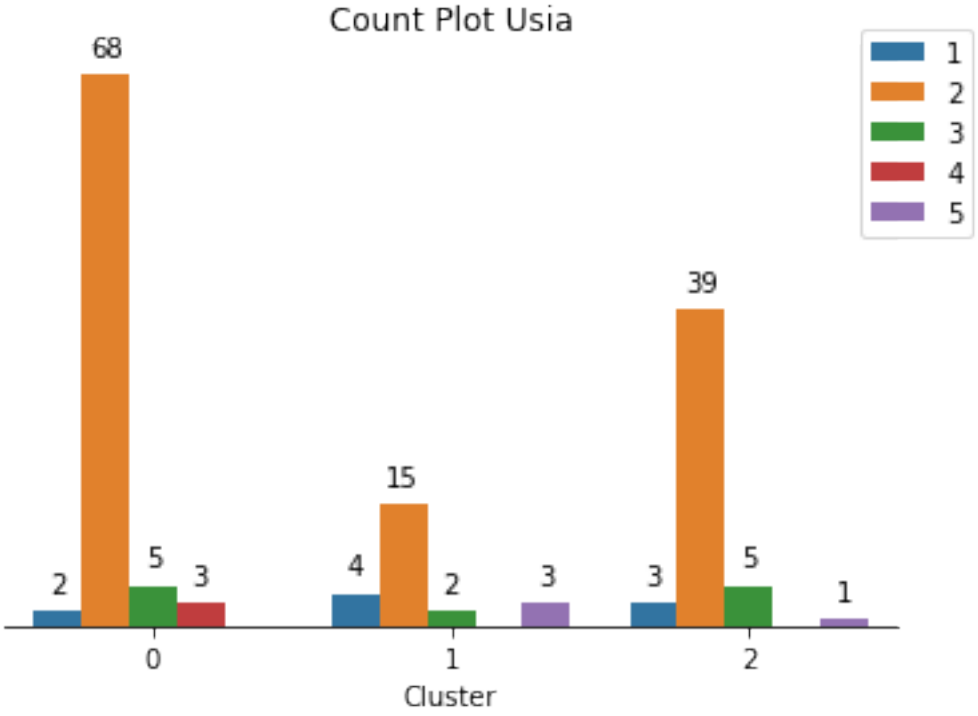
	Frekuensi Pembelian	TRU1	TRU2	TRU3	TRU4	…	FM	FCS	FTH	TMF	FD	\
0	3	4	5	4	4	…	5	2	5	4	2	
1	6	4	5	5	4	…	5	5	5	5	5	
2	7	4	3	3	4	…	3	3	5	3	5	
3	3	4	5	4	4	…	5	5	4	4	5	
4	7	5	5	5	5	…	5	5	5	5	5	

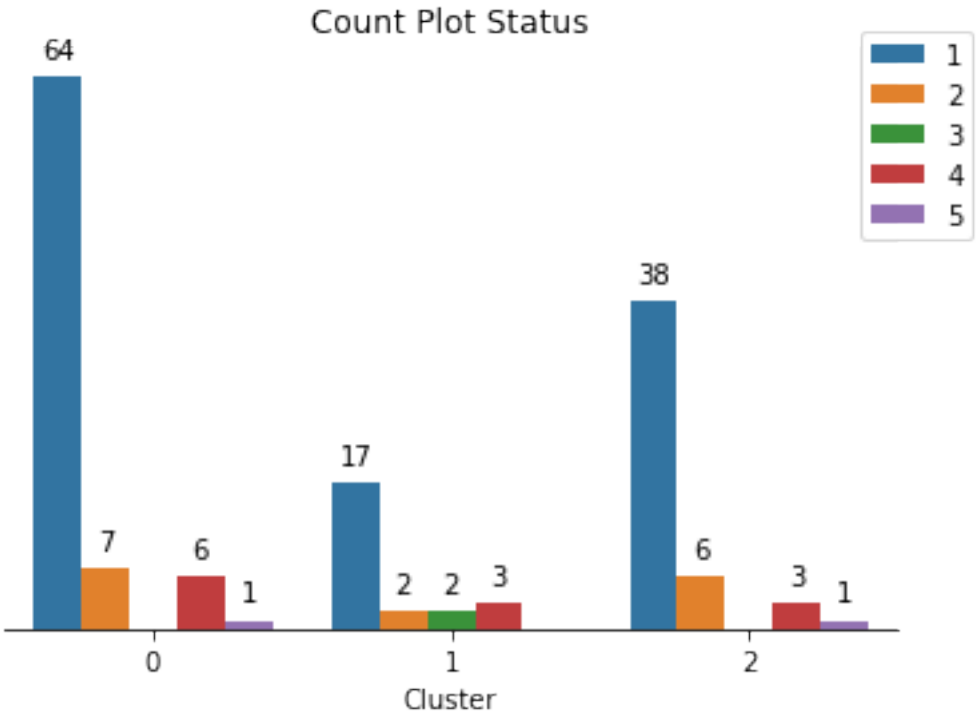
	REL1	REL2	REL3	REL4	Cluster
0	4	5	2	4	2
1	5	2	5	5	0
2	4	4	5	5	0
3	4	4	4	5	2
4	5	5	5	5	0

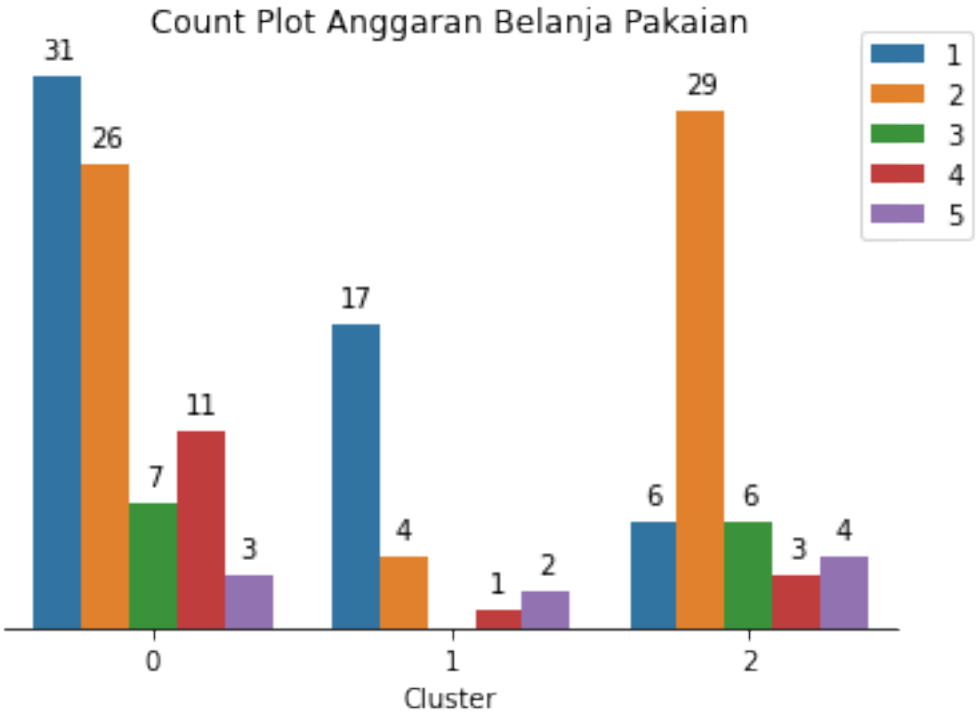
[5 rows x 24 columns]

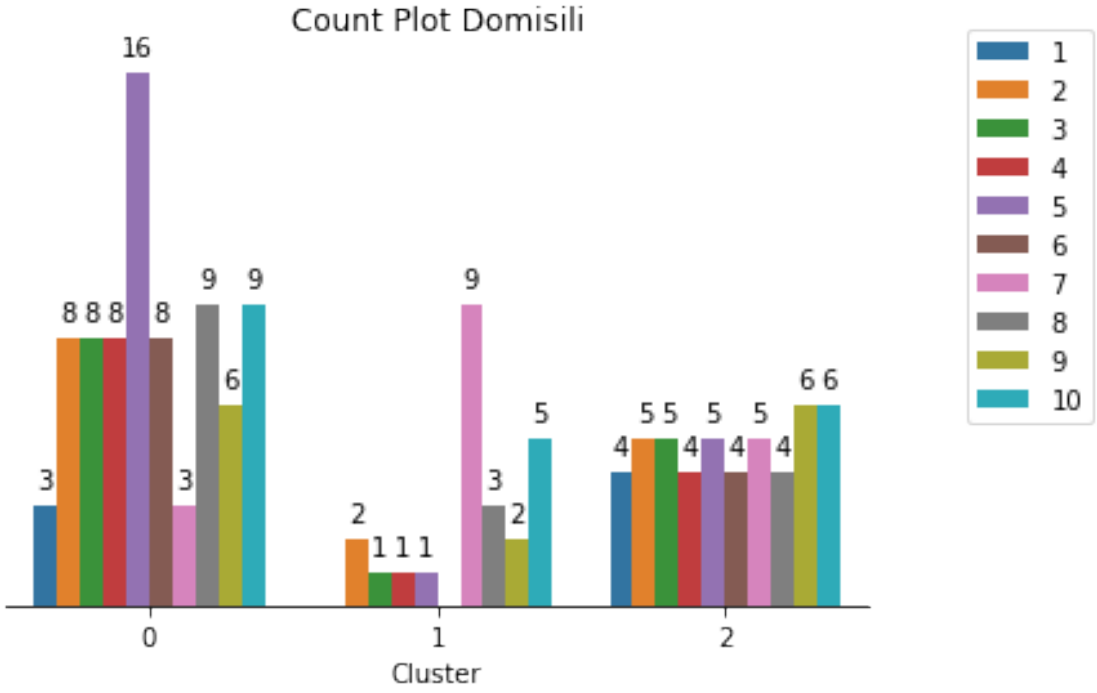
[43]:
```
# Visualisasi cluster yang terbentuk
for i in df:
  plt.figure(figsize=(6,4))
  ax=sns.countplot(data=df_cluster, x='Cluster', hue=i)
  plt.title('\nCount Plot {}'.format(i), fontsize=12)
  ax.legend(loc='best')
  ax.legend(bbox_to_anchor=(1.1, 1.05))
  for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'),
    (p.get_x() + p.get_width()/2., p.get_height()),
    ha='center',
    va='center',
    xytext=(0,10),
    textcoords = 'offset points')
  sns.despine(right=True, top=True, left=True)
  ax.axes.yaxis.set_visible(False)
  plt.show()
```

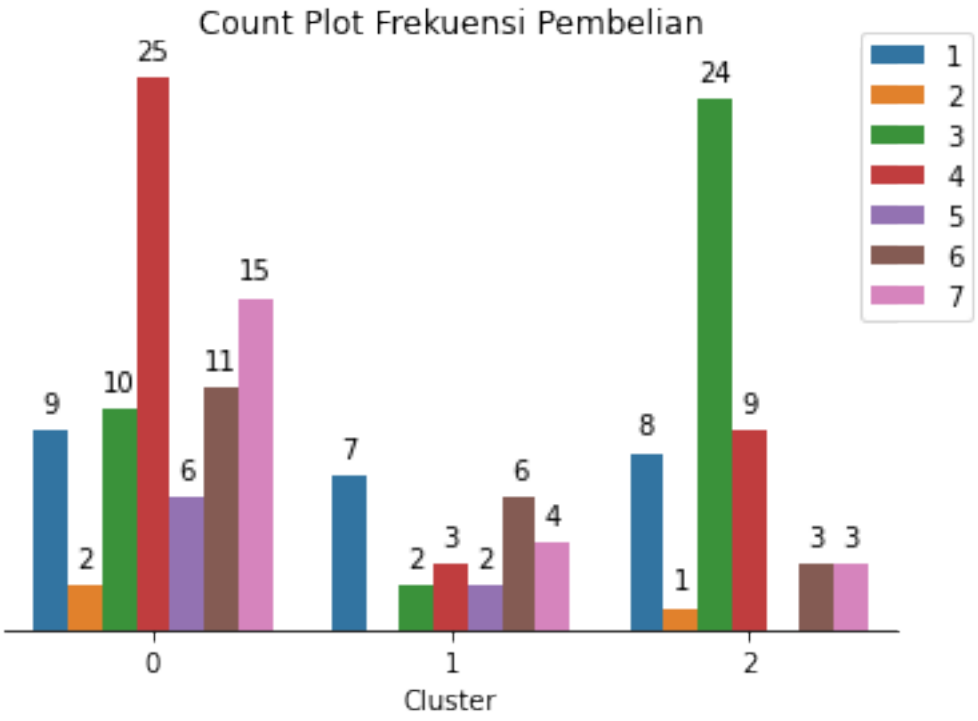

Count Plot Jenis Kelamin
1
2
52
26
16
8
12
36
0
1
2
Cluster

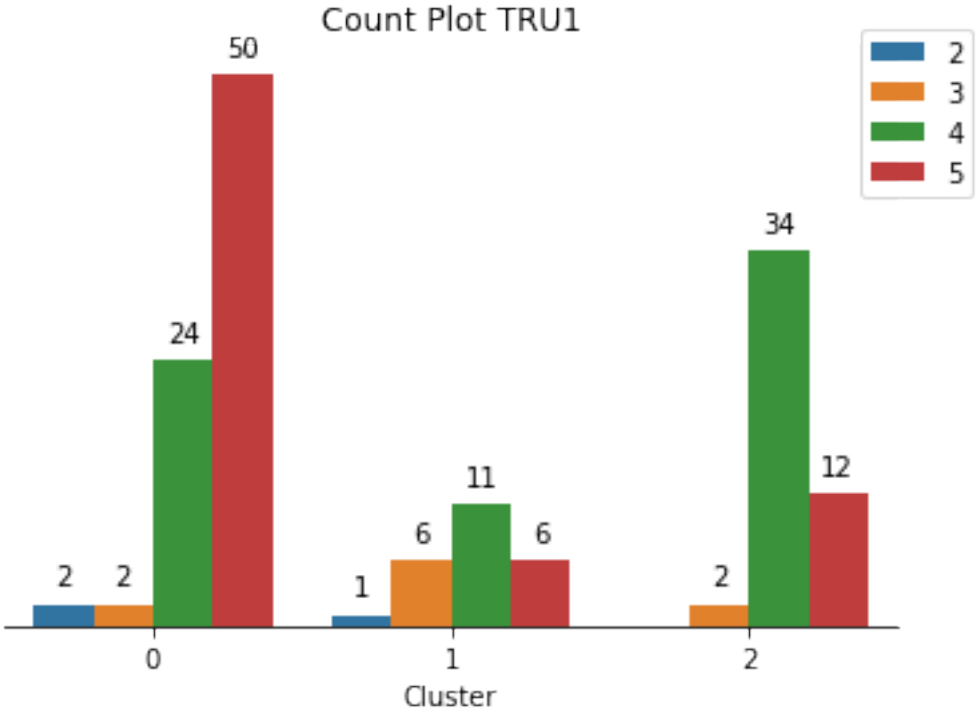

Count Plot Usia
1
2
3
4
5
68
2
5
3
15
4
2
3
39
3
5
1
0
1
2
Cluster

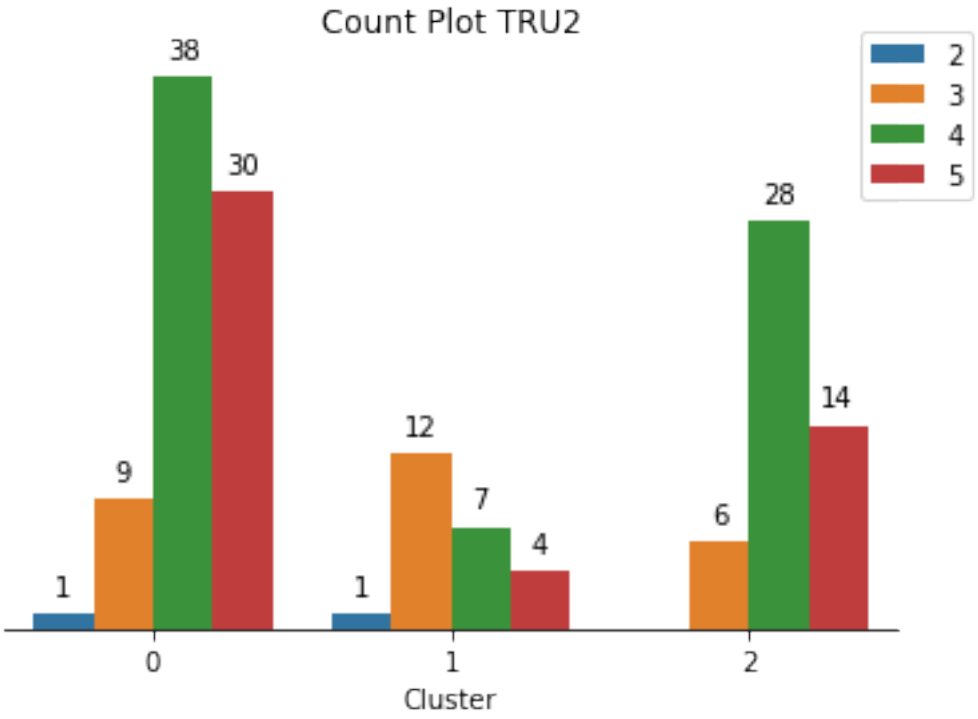

Count Plot Status
64
7
6
1
17
2
2
3
38
6
3
1
0
1
2
Cluster
1
2
3
4
5

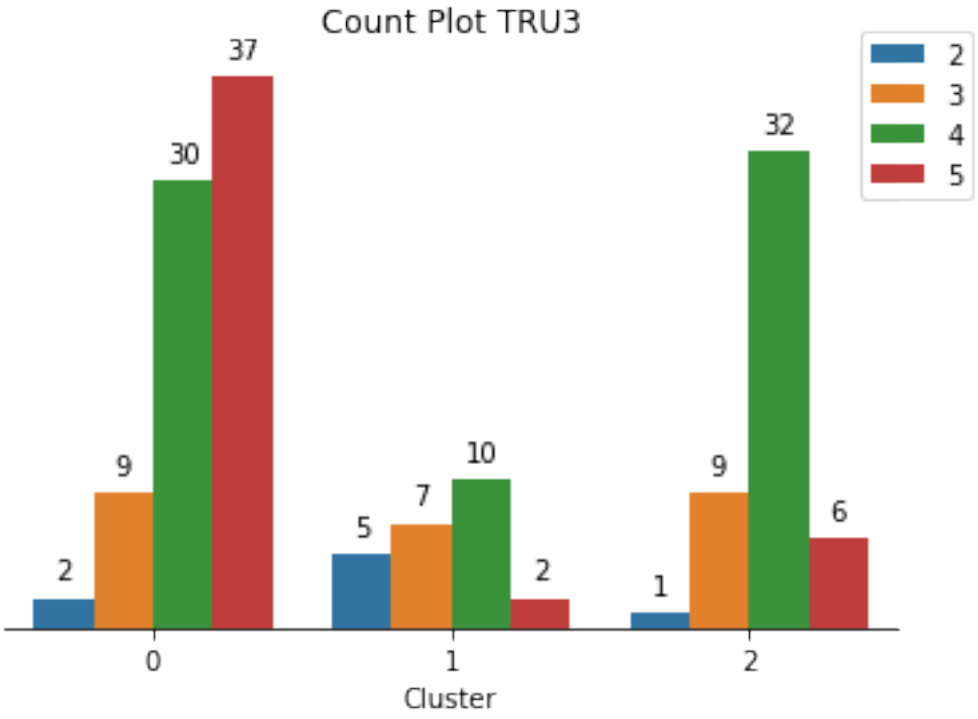

Count Plot Anggaran Belanja Pakaian
31
26
7
11
3
17
4
1
2
6
29
6
3
4
0
1
2
Cluster
1
2
3
4
5

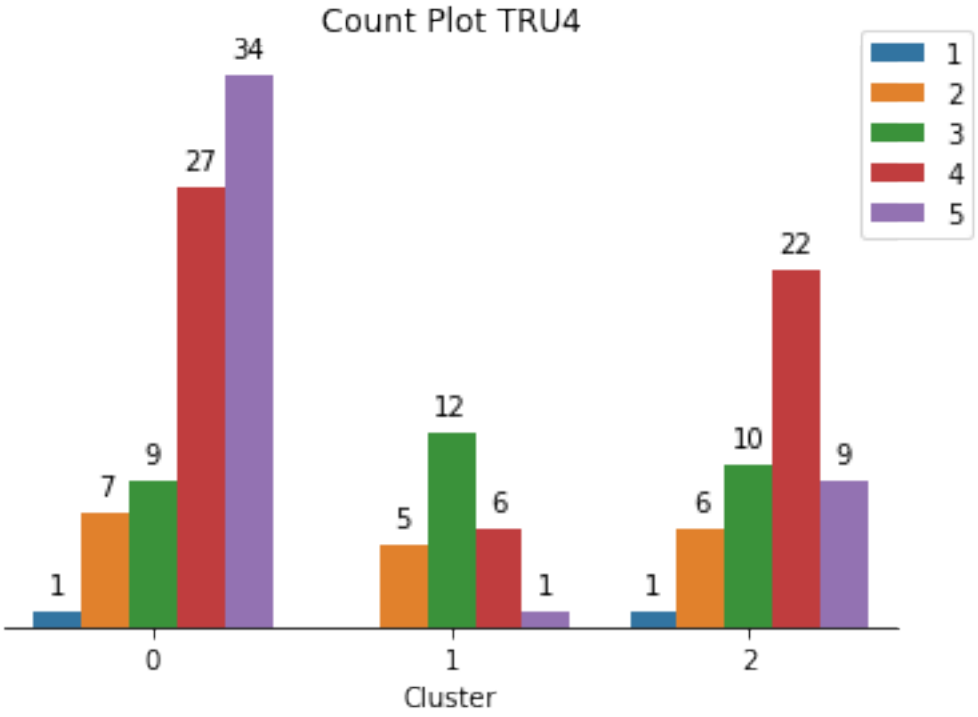

Count Plot Domisili
16
8 8 8 8
9 9
6
3
3
2
1 1 1
9
3
2
5
4
5 5
4
5
4
5
4
6 6
0
1
2
Cluster
1
2
3
4
5
6
7
8
9
10

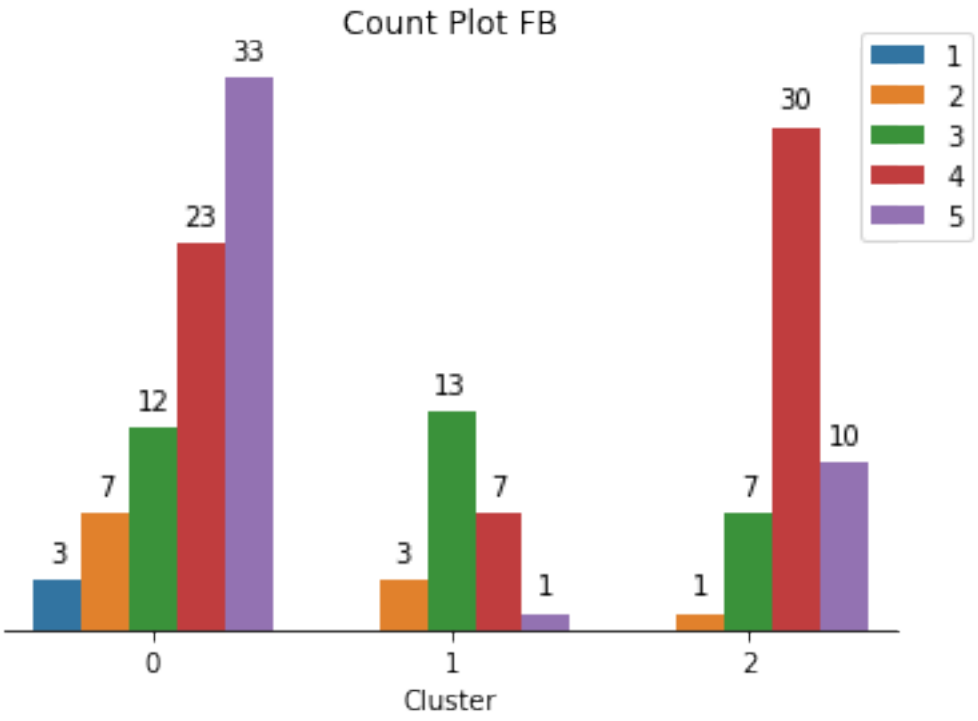

Count Plot Frekuensi Pembelian
1
2
3
4
5
6
7
9
2
10
25
6
11
15
7
2
3
2
6
4
8
1
24
9
3
3
0
1
2
Cluster

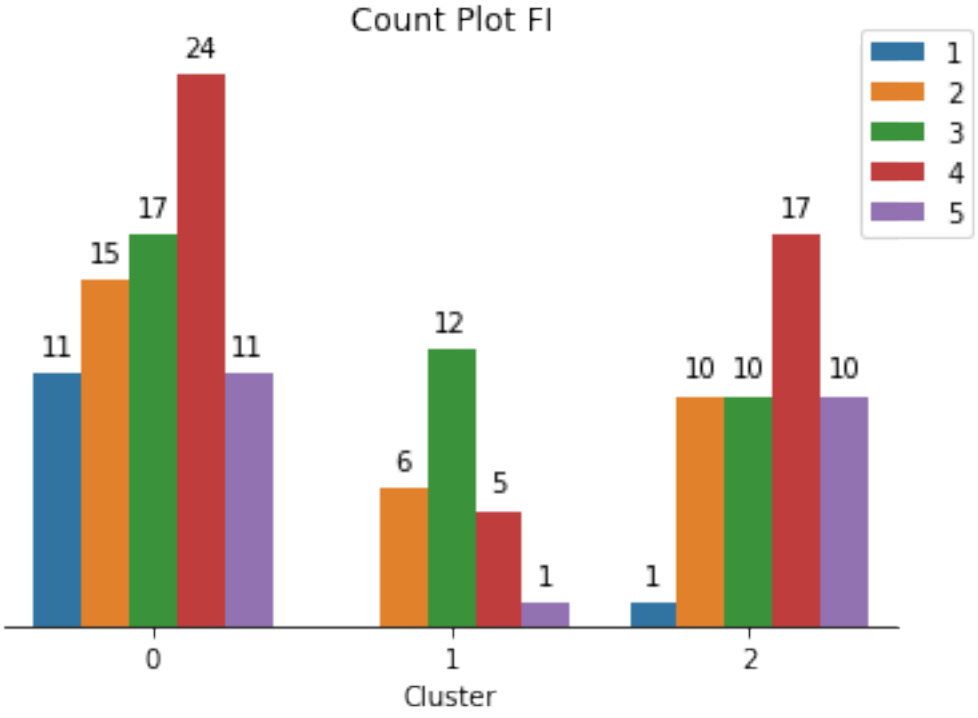

Count Plot TRU1
2
3
4
5
2
2
24
50
1
6
11
6
2
34
12
0
1
2
Cluster

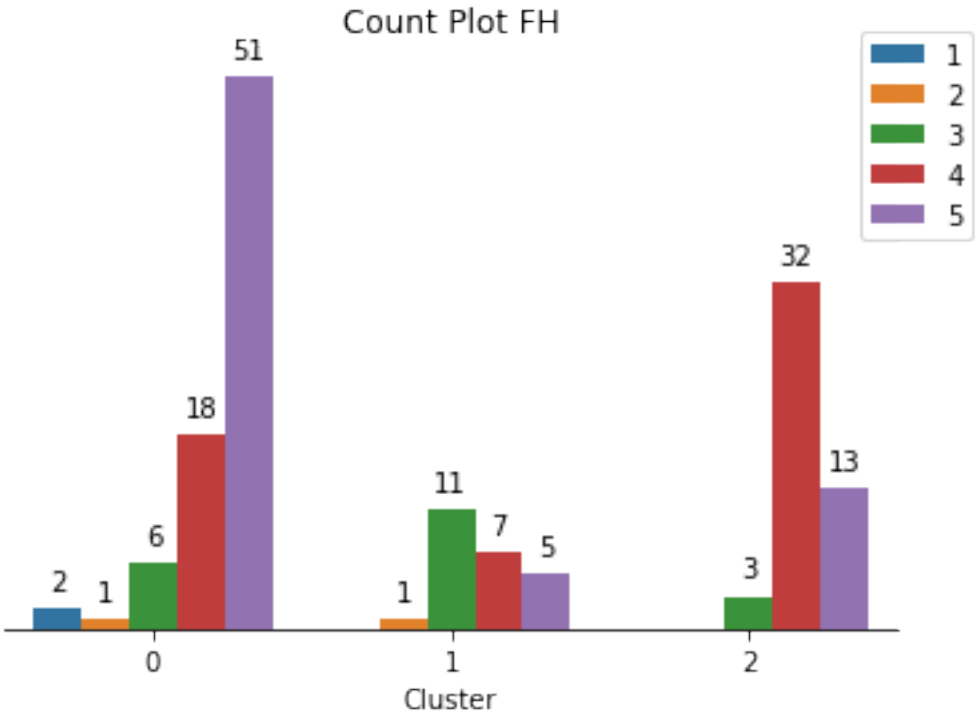

Count Plot TRU2
2
3
4
5
38
30
9
1
12
7
4
1
28
14
6
0
1
2
Cluster

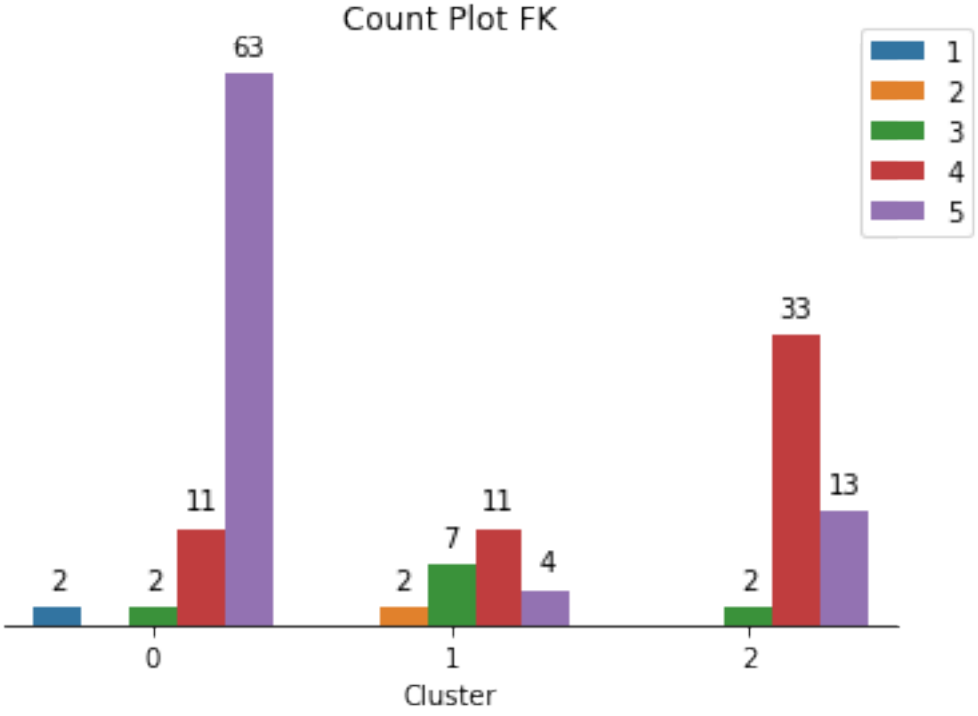

Count Plot TRU3
2
3
4
5
37
30
9
2
10
7
5
2
32
9
6
1
0
1
2
Cluster

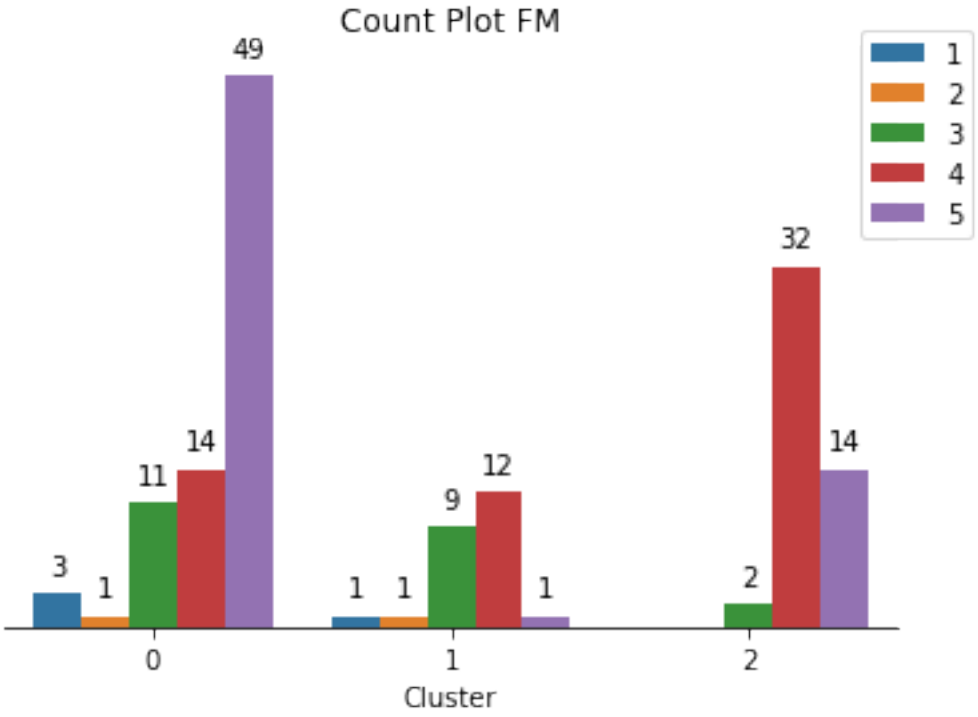

Count Plot TRU4
1
2
3
4
5
34
27
9
7
1
12
6
5
1
22
10
6
1
9
0
1
2
Cluster

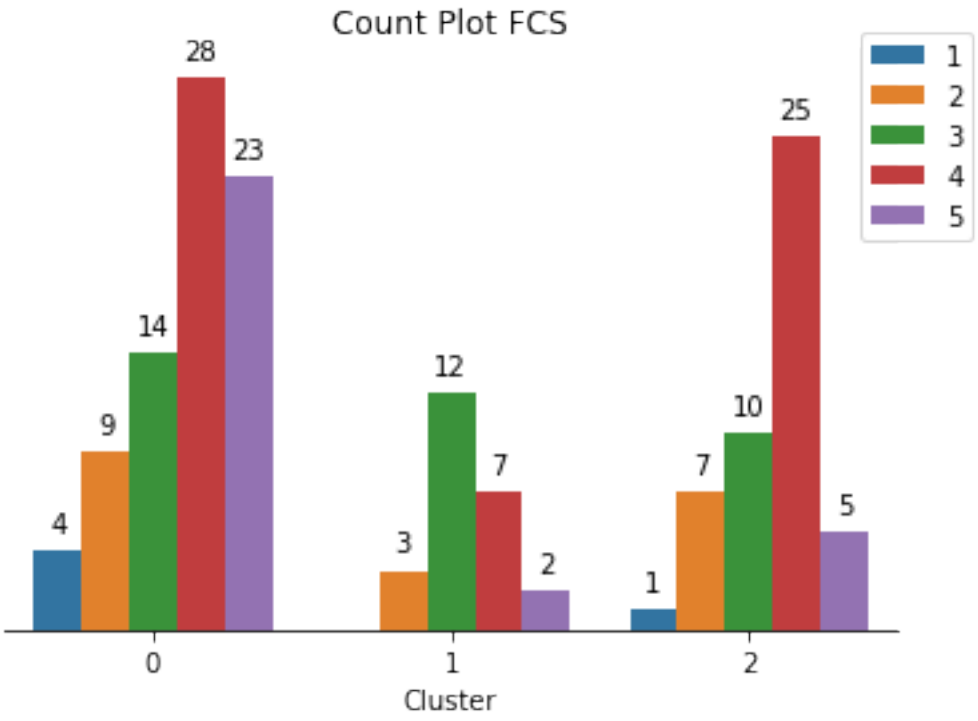

Count Plot FB
1
2
3
4
5
33
23
12
7
3
13
7
3
1
30
7
1
10
0
1
2
Cluster

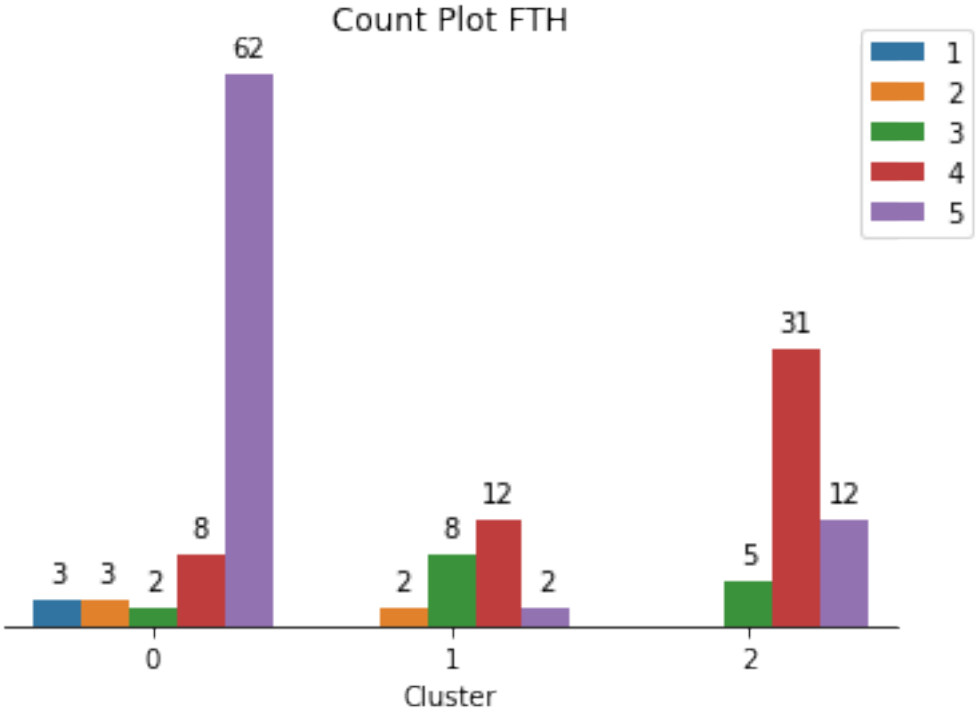

Count Plot FI
1
2
3
4
5
24
17
15
11
11
12
6
5
1
17
10
10
1
10
0
1
2
Cluster

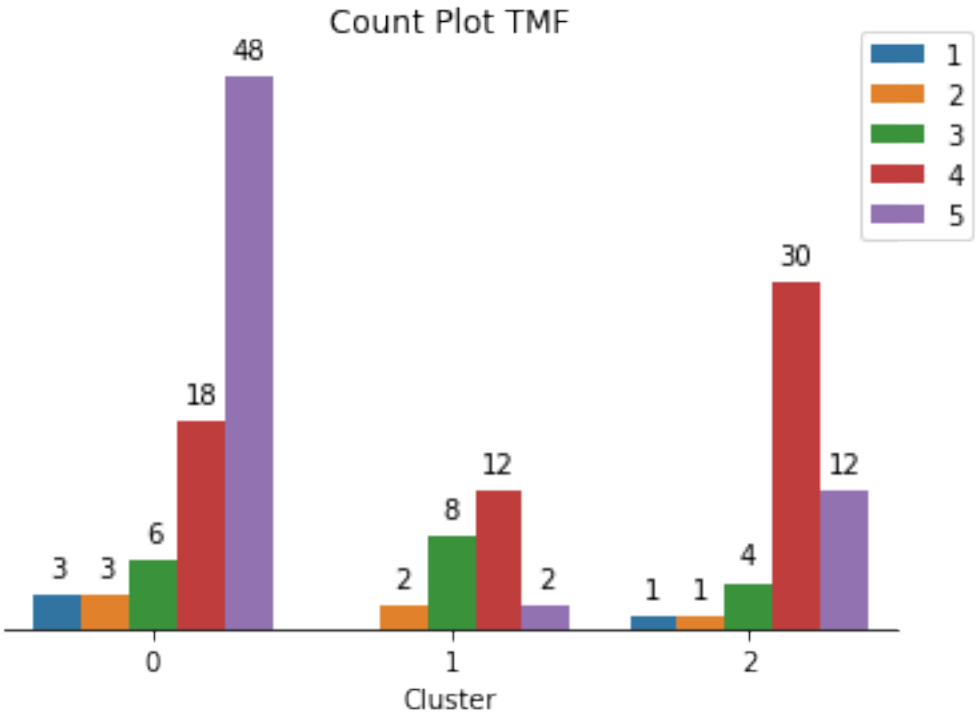

Count Plot FH
1
2
3
4
5
51
18
6
2
1
11
7
5
1
32
13
3
0
1
2
Cluster

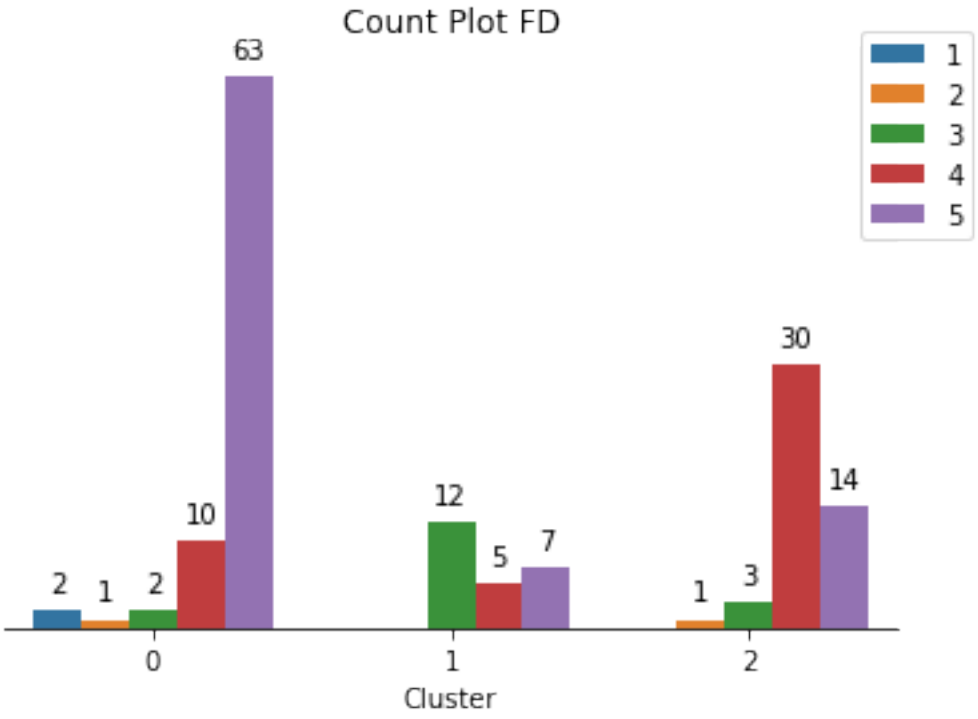

Count Plot FK
1
2
3
4
5
63
11
2
2
11
7
4
2
33
13
2
0
1
2
Cluster

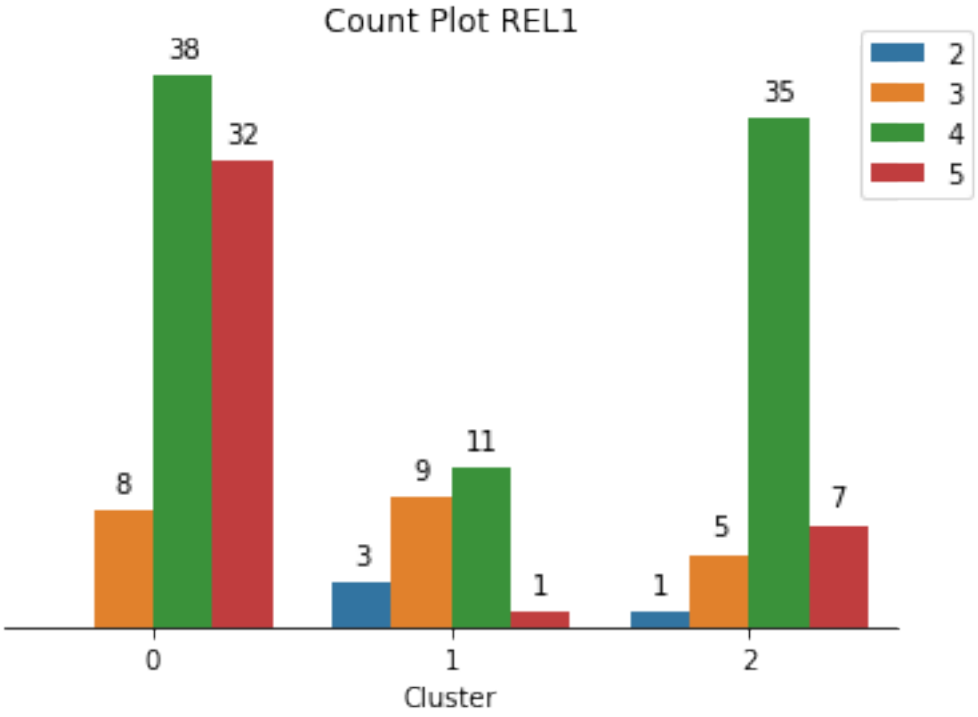

Count Plot FM
1
2
3
4
5
49
14
11
3
1
12
9
1
1
1
32
14
2
0
1
2
Cluster

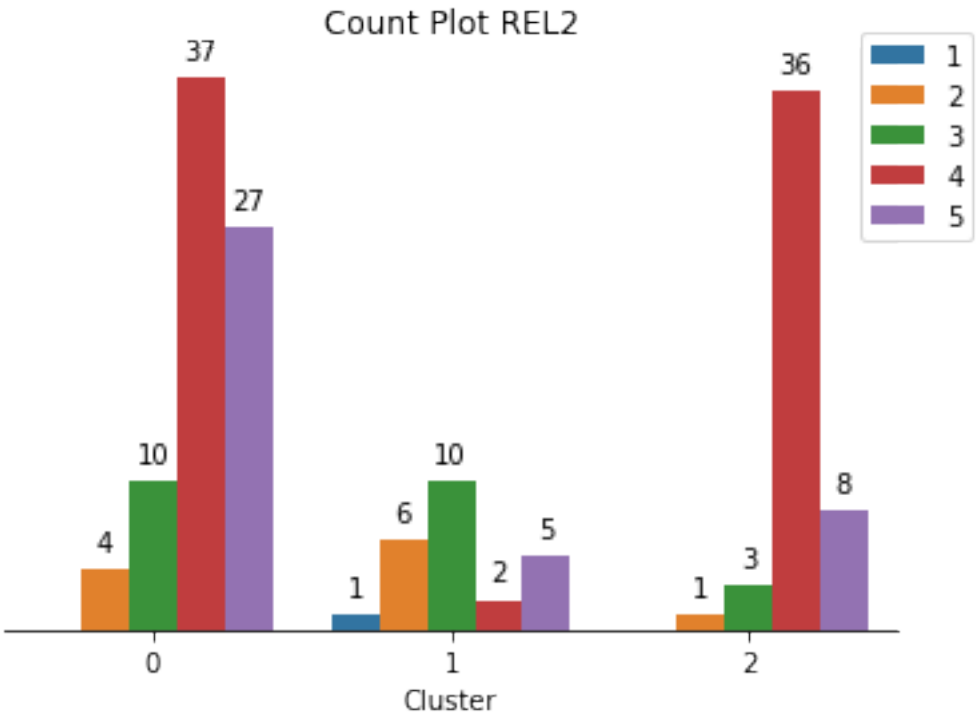

Count Plot FCS
1
2
3
4
5
28
23
14
9
4
12
7
3
2
25
10
7
1
5
0
1
2
Cluster

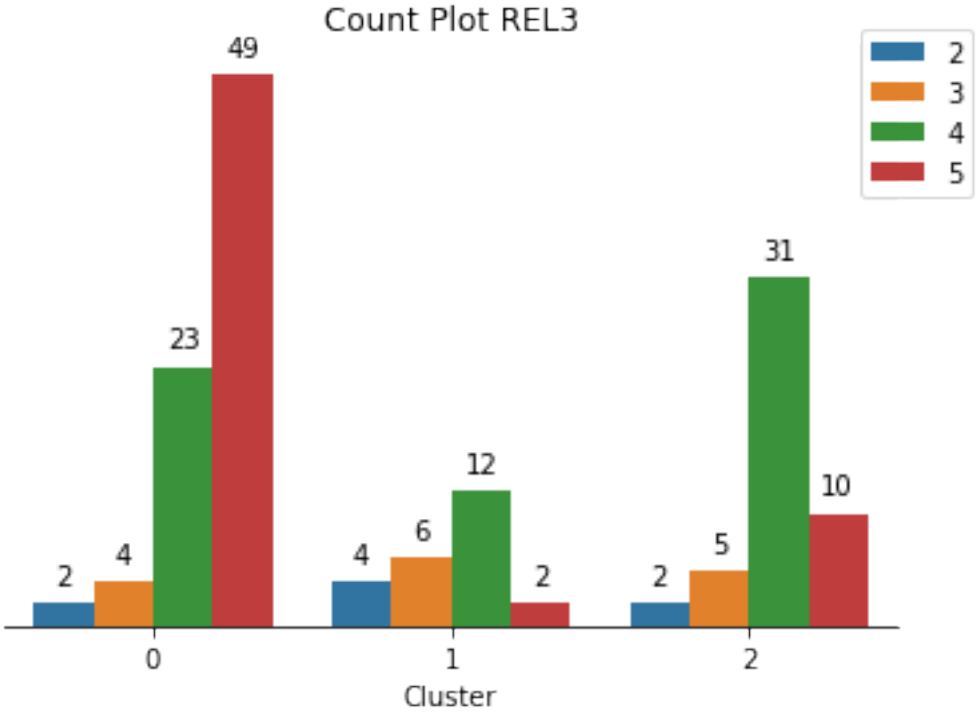

Count Plot FTH
1
2
3
4
5
62
8
3
3
2
12
8
2
2
31
12
5
0
1
2
Cluster

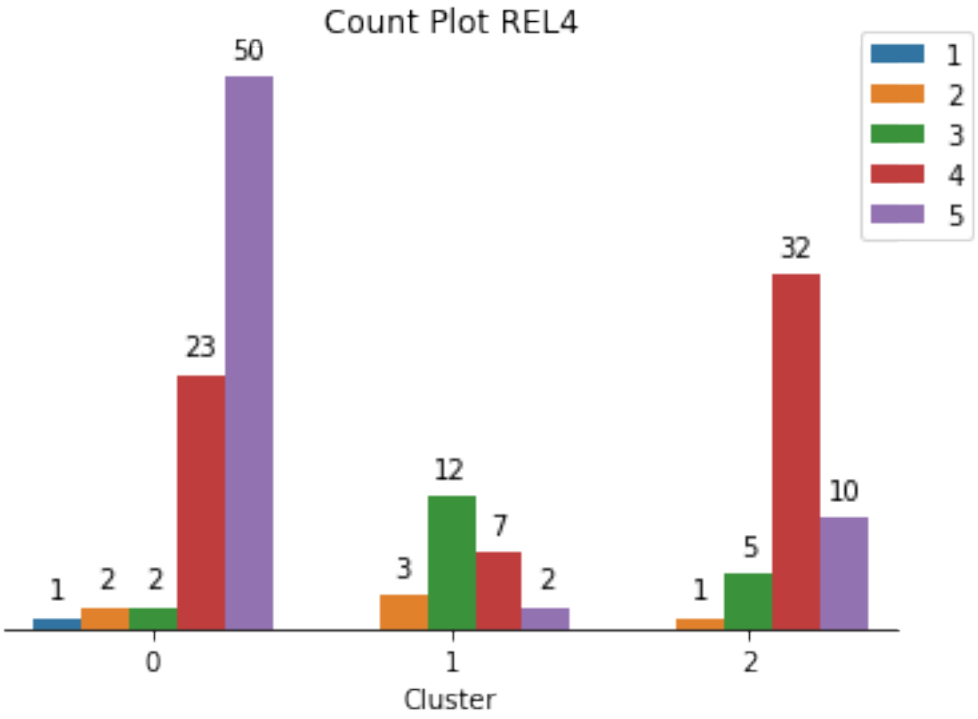

Count Plot TMF
1
2
3
4
5
48
18
6
3
3
12
8
2
2
30
12
4
1
1
0
1
2
Cluster


Count Plot FD
1
2
3
4
5
63
10
2
1
2
12
5
7
30
14
3
1
0
1
2
Cluster


Count Plot REL1
2
3
4
5
38
32
8
3
9
11
1
1
5
35
7
0
1
2
Cluster


Count Plot REL2
1
2
3
4
5
37
27
10
4
10
6
1
2
5
36
8
3
1
0
1
2
Cluster


Count Plot REL3
2
3
4
5
49
23
4
2
12
6
4
2
31
10
5
2
0
1
2
Cluster


Count Plot REL4
1
2
3
4
5
50
23
2
2
1
12
7
3
2
32
5
1
10
0
1
2
Cluster

[ ]: