Unsupervised Learning Project: AllLife Bank Customer Segmentation

---

Marks: 30

Welcome to the project on Unsupervised Learning. We will be using **Credit Card Customer Data** for this project.

Context

AllLife Bank wants to focus on its credit card customer base in the next financial year. They have been advised by their marketing research team, that the penetration in the market can be improved. Based on this input, the marketing team proposes to run personalized campaigns to target new customers as well as upsell to existing customers.

Another insight from the market research was that the customers perceive the support services of the bank poorly. Based on this, the operations team wants to upgrade the service delivery model, to ensure that customers' queries are resolved faster. The head of marketing and the head of delivery, both decide to reach out to the Data Science team for help.

Objective

Identify different segments in the existing customer base, taking into account their spending patterns as well as past interactions with the bank.

About the data

Data is available on customers of the bank with their credit limit, the total number of credit cards the customer has, and different channels through which the customer has contacted the bank for any queries. These different channels include visiting the bank, online, and through a call center.

- **Sl_no** - Customer Serial Number
- **Customer Key** - Customer identification
- **Avg_Credit_Limit** - Average credit limit (currency is not specified, you can make an assumption around this)
- **Total_Credit_Cards** - Total number of credit cards
- **Total_visits_bank** - Total bank visits
- **Total_visits_online** - Total online visits
- **Total_calls_made** - Total calls made

Importing libraries and overview of the dataset

In [112…

```
# Importing all the necessary packages

import pandas as pd

import numpy as np

import matplotlib.pylab as plt

import seaborn as sns

# To scale the data using z-score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# Importing clustering algorithms
from sklearn.cluster import KMeans

from sklearn.mixture import GaussianMixture

from sklearn_extra.cluster import KMedoids

import warnings
warnings.filterwarnings("ignore")
```

Loading the data

In [46]:

```
bank = pd.read_excel('Credit+Card+Customer+Data.xlsx')
```

In [47]:
```
bank.head()
```

Out[47]:

	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online
0	1	87073	100000	2	1	1
1	2	38414	50000	3	0	10
2	3	17341	50000	7	1	3
3	4	40496	30000	5	1	1
4	5	47437	100000	6	0	12

In [48]:
```
bank.sample(10, random_state = 10)
```

Out[48]:

	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_onl
254	255	23302	16000	4	3	
349	350	11799	11000	7	3	
295	296	41380	10000	6	4	
35	36	30888	19000	2	0	
377	378	61994	19000	5	2	
484	485	29102	28000	5	4	
257	258	21531	10000	6	4	
78	79	59656	6000	2	0	
386	387	85122	18000	5	2	
285	286	73952	11000	5	5	

In [49]:
```
bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 660 entries, 0 to 659
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Sl_No                660 non-null    int64
 1   Customer Key         660 non-null    int64
 2   Avg_Credit_Limit     660 non-null    int64
 3   Total_Credit_Cards   660 non-null    int64
 4   Total_visits_bank    660 non-null    int64
 5   Total_visits_online  660 non-null    int64
 6   Total_calls_made     660 non-null    int64
dtypes: int64(7)
memory usage: 36.2 KB
```

In [50]:
```
bank.shape
```

Out[50]:
```
(660, 7)
```

In [51]:
```
bank.nunique()
```

Out[51]:
```
Sl_No                  660
Customer Key           655
Avg_Credit_Limit       110
Total_Credit_Cards      10
Total_visits_bank        6
Total_visits_online     16
Total_calls_made        11
dtype: int64
```

Data Overview:

- There are **660 entries with 7 columns**
- All 660 columns have non-null values, **there are no missing values.**
- All columns are **integer type**

Data Preprocessing and Exploratory Data Analysis

Cleaning the data/ Cheking for duplicates

In [53]:
```
# Check for duplicates:

duplicate_rows = bank[bank.duplicated('Customer Key')]

duplicate_rows
```

Out[53]:

	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_onl
332	333	47437	17000	7	3	
398	399	96929	67000	6	2	
432	433	37252	59000	6	2	
541	542	50706	60000	7	5	
632	633	97935	187000	7	1	

In [54]:
```
bank.drop_duplicates()
```

Out[54]:

	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_onl
0	1	87073	100000	2	1	
1	2	38414	50000	3	0	
2	3	17341	50000	7	1	
3	4	40496	30000	5	1	
4	5	47437	100000	6	0	
...	...	...	...	...	...	
655	656	51108	99000	10	1	
656	657	60732	84000	10	1	
657	658	53834	145000	8	1	
658	659	80655	172000	10	1	
659	660	80150	167000	9	0	

660 rows × 7 columns

In [55]:
```
bank.shape
```

Out[55]:
```
(660, 7)
```

In [56]:
```
bank.drop(columns = ['Customer Key','Sl_No'], inplace = True)
```

In [57]:
```
bank[bank.duplicated()]
```

Out[57]:

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_ma
162	8000	2	0	3	
175	6000	1	0	2	
215	8000	4	0	4	
295	10000	6	4	2	
324	9000	4	5	0	
361	18000	6	3	1	
378	12000	6	5	2	
385	8000	7	4	2	
395	5000	4	5	0	
455	47000	6	2	0	
497	52000	4	2	1	

In [58]:
```
bank = bank[~bank.duplicated()]
```

In [59]:
```
bank.shape
```

Out[59]:
```
(649, 5)
```

Cleaning:

- There were 5 **duplicated values on the Customer Key column** as it shows only 655 unique values.
- **Removed the duplicate values.**
- **Dropped the S1_No and Customer Key columns** as they are not required for the analysis.

Check the summary Statistics

In [60]:
```
bank.describe().T
```

Out[60]:

	count	mean	std	min	25%	50%	75%	
Avg_Credit_Limit	649.0	34878.274268	37813.736638	3000.0	11000.0	18000.0	49000.0	200
Total_Credit_Cards	649.0	4.708783	2.173763	1.0	3.0	5.0	6.0	
Total_visits_bank	649.0	2.397535	1.625148	0.0	1.0	2.0	4.0	
Total_visits_online	649.0	2.624037	2.952888	0.0	1.0	2.0	4.0	
Total_calls_made	649.0	3.590139	2.877911	0.0	1.0	3.0	5.0	

Observations:

649 customers in the dataset:

Avg_Credit_Limit:

- The average credit limit 34,878.27.
- The std is high indicating a wide range of credit limits
- The minimum credit limit is 3,000, while the maximum is 200,000.
- The median credit limit is 18,000, which suggests that half of the individuals have a credit limit below this amount.
- The majority of individuals have credit limits between 11,000 and 49,000.

Total_Credit_Cards:

- On average, individuals in the dataset possess around 4.71 credit cards.
- The std is 2.17, indicating some variability in the number of credit cards number.
- The minimum number of credit cards is 1 the maximum is 10.
- The median value is 5, suggesting that half of the individuals have 5 or fewer credit cards.
- The majority of individuals have between 3 and 6 credit cards.

Total_visits_bank:

- On average, individuals make approximately 2.40 visits to a bank.
- The std ia 1.63, indicating some variability in the frequency of bank visits.
- The minimum number of bank visits is 0, while the maximum is 5.
- The median value is 2, indicating that half of the individuals make 2 or fewer bank visits.
- The majority of individuals visit the bank between 1 and 4 times.

Total_visits_online:

- On average, individuals make around 2.62 visits to online platforms.
- The standard deviation is 2.95, suggesting a significant variation in online visitation patterns.
- The minimum number of online visits is 0, while the maximum is 15.
- The median value is 2, implying that half of the individuals make 2 or fewer online visits.
- The majority of individuals make between 1 and 4 online visits.

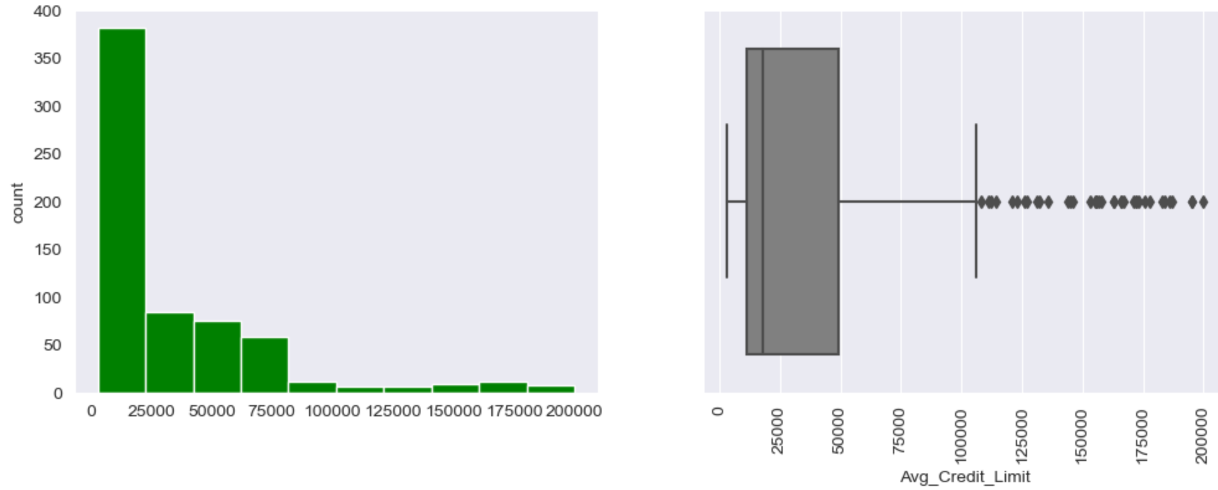
Total_calls_made:

- On average, individuals make approximately 3.59 calls.
- The standard deviation is 2.88, indicating some variability in the number of calls made.
- The minimum number of calls made is 0, while the maximum is 10.
- The median value is 3, suggesting that half of the individuals make 3 or fewer calls.
- The majority of individuals make between 1 and 5 calls.

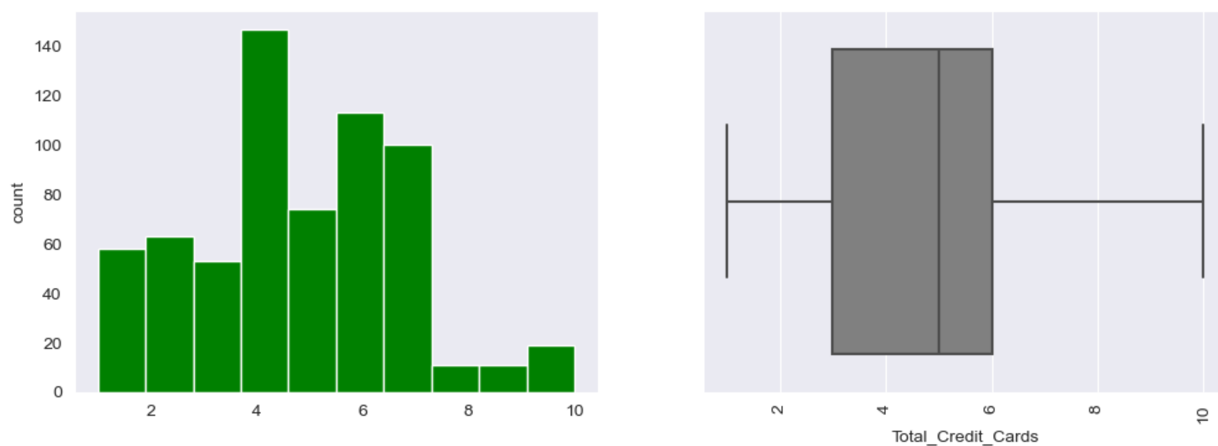
In [118…

```
num_cols = list(bank.columns)

for col in num_cols:

    print(col)

    print('Skew :',round(bank[col].skew(),2))

    plt.figure(figsize = (12, 4))

    plt.subplot(1, 2, 1)

    bank[col].hist(bins = 10, grid = False, color = 'green')

    plt.ylabel('count')

    plt.subplot(1, 2, 2)

    sns.boxplot(x = bank[col], color = 'gray')

    plt.xticks(rotation = 90)

    plt.show()
```

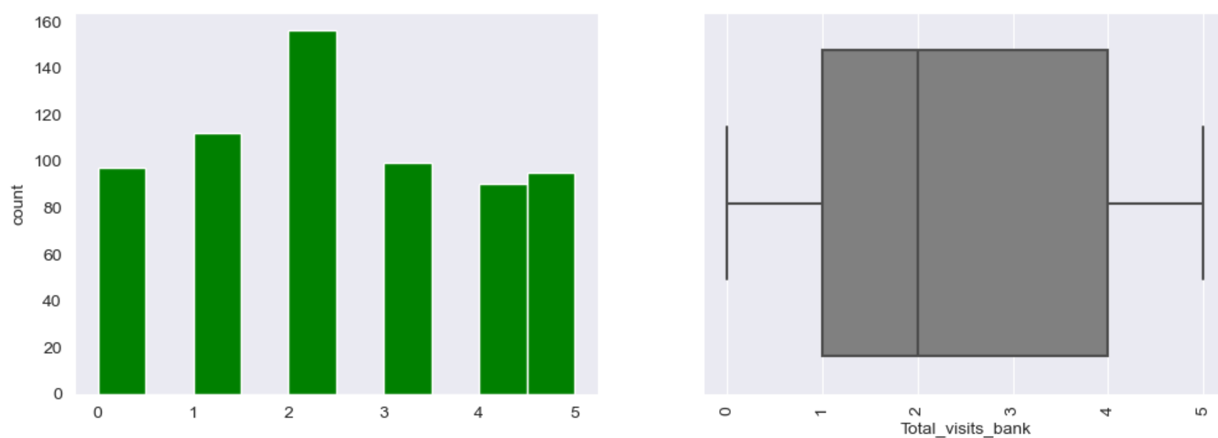
```
Avg_Credit_Limit
Skew : 2.19
```



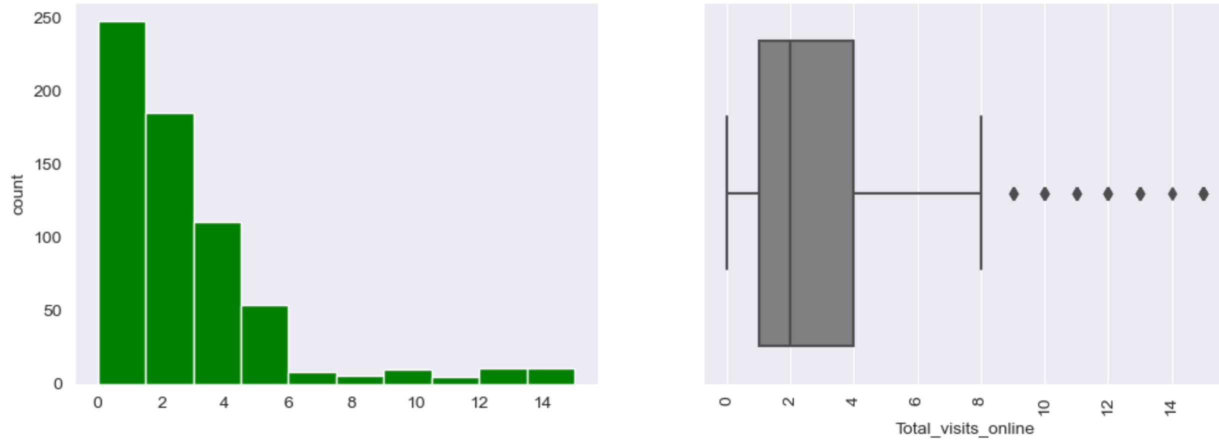

```
Total_Credit_Cards
Skew : 0.15
```



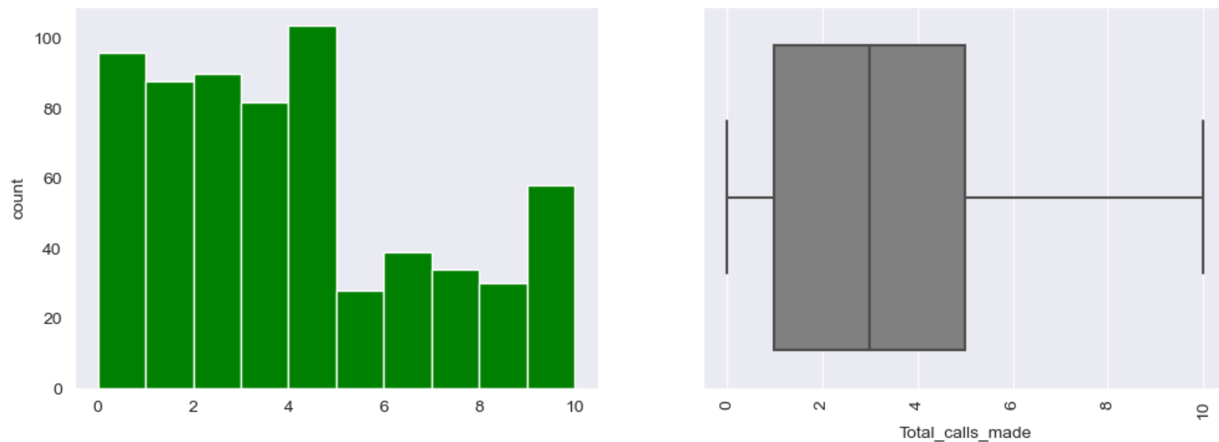

```
Total_visits_bank
Skew : 0.15
```



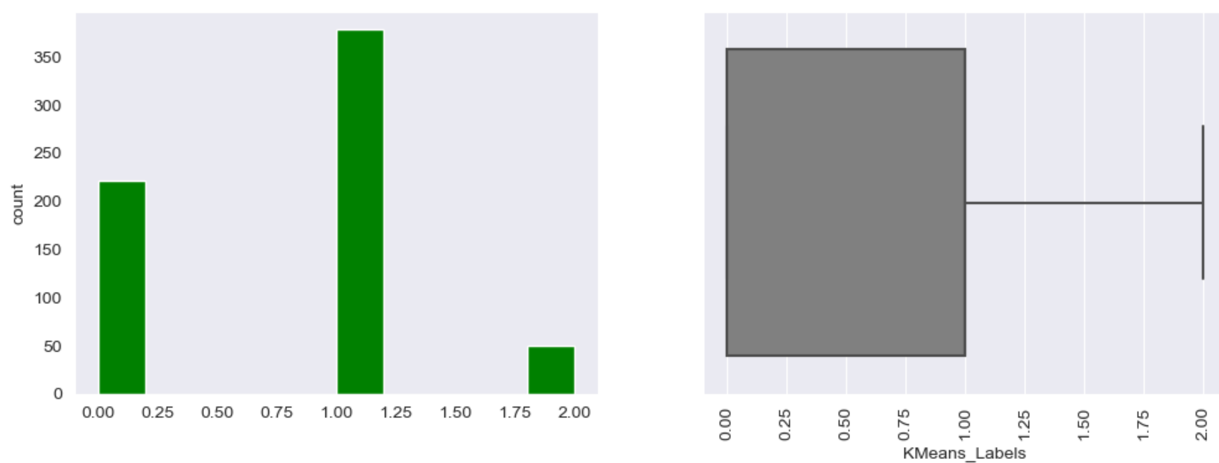

```
Total_visits_online
Skew : 2.21
```



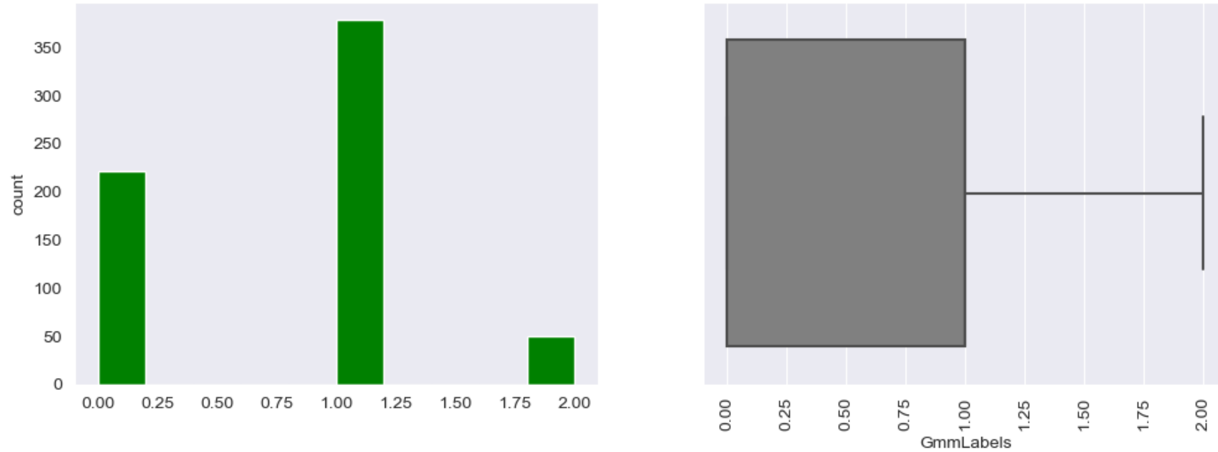

```
Total_calls_made
Skew : 0.66
```



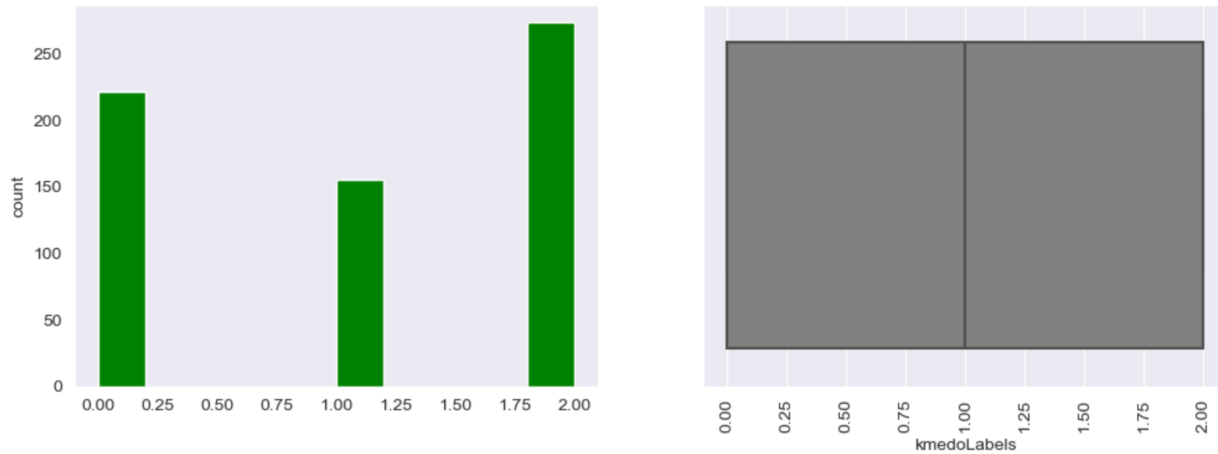

```
KMeans_Labels
Skew : 0.15
```



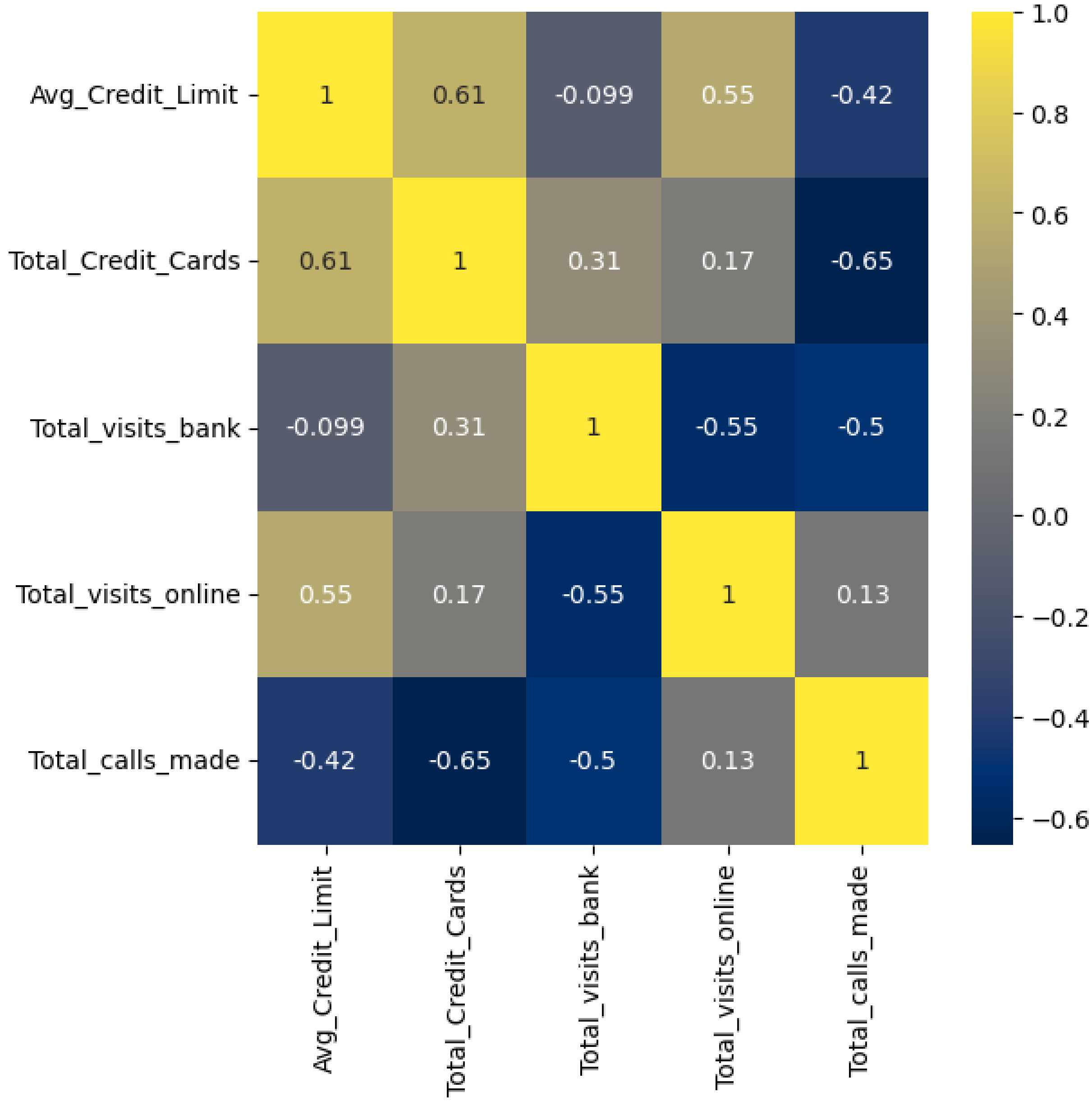

```
GmmLabels
Skew : 0.15
```




```
kmedoLabels
Skew : -0.16
```



In [76]:
```python
plt.figure(figsize = (6, 6))

sns.heatmap(bank.corr(), annot = True, cmap = "cividis")

plt.xticks(rotation = 90)

plt.show()
```



Observations:

Distribution and Outliers plots show:

- Avg_Credit_Limit: left skew distribution and plenty of outliers
- Total_Credit_Cards: normal distribution and no outliers
- Total_visits_bank: moderate normal distribution and no outliers
- Total_visits_online: left skew distribution and moderate numbers of outliers
- Total_calls_made: left distribution and no outliers

Correlation map:

moderate positive correlation:

- average credit limit and the total credit cards (0.61)
- total visits online and the average credit limit (0.55)

weak negative correlation:

- total visits bank and total calls made (-0.526)
- total visits online and total calls made (-0.436)

weak positive correlation:

- total credit cards and total visits to the bank (0.31)

Average credit limit and total credit cards seem moderately correlated. Number of calls and total credit cards seem not related.

Scaling

In [77]:
```
# Scaling cars
scaler = StandardScaler()

bank_scaled = pd.DataFrame(scaler.fit_transform(bank), columns = bank.columns)
```

In [78]:
```
bank_scaled.head(2)
```

Out[78]:

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
0	1.723499	-1.247087	-0.860606	-0.550407	-1.248443
1	0.400209	-0.786701	-1.476410	2.499808	1.881237

Apply PCA to scaled data

In [79]:
```
from sklearn.decomposition import PCA

n = bank.shape[1]

# Create a PCA instance: pca
pca = PCA(n_components=n)

principal_components = pca.fit_transform(bank_scaled)

# Save components to a DataFrame
bank_pca = pd.DataFrame(principal_components, columns = bank.columns)
```

In [80]:
```
bank_copy = bank_pca.copy(deep = True)
```

K-Means

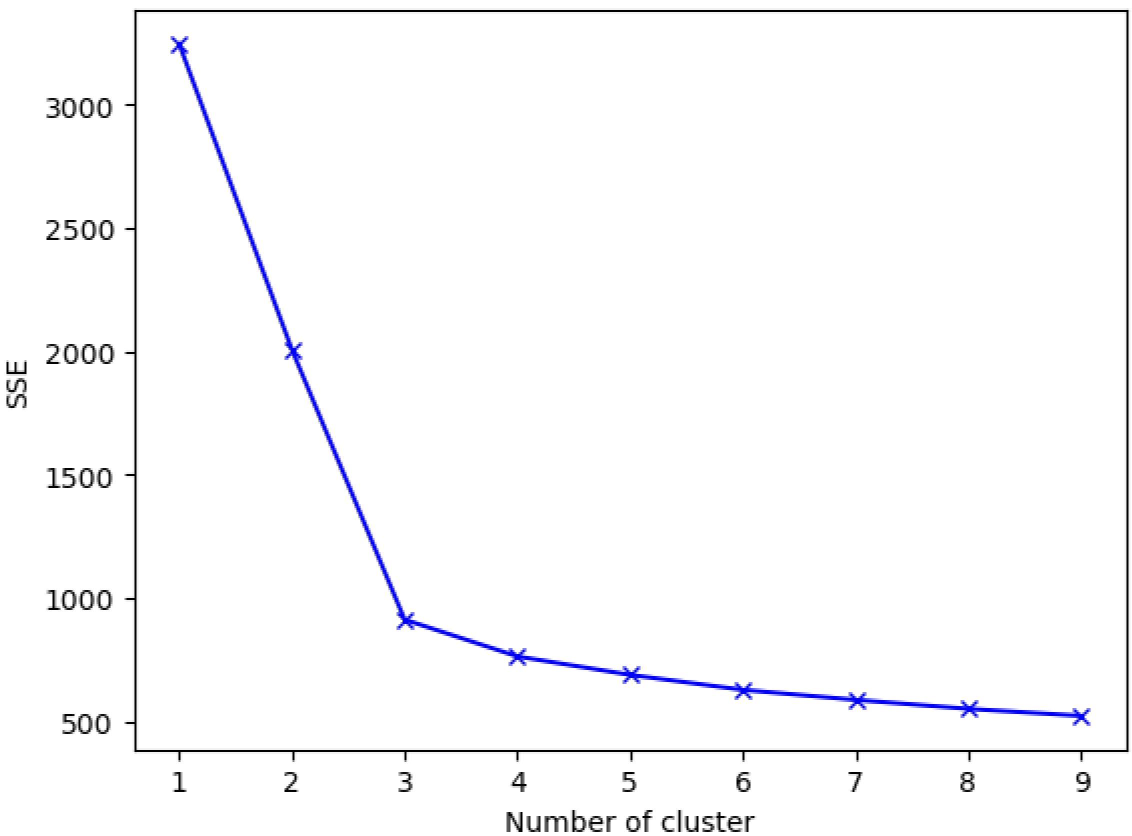
Let us now fit the K-means algorithm on our pca components and find out the optimum number of clusters to use.

We will do this in 3 steps:

1. Initialize a dictionary to store the Sum of Squared Error (SSE) for each K

2. Run for a range of Ks and store SSE for each run
3. Plot the SSE vs K and plot the elbow curve

In [81]:
```
# step 1
sse = {}

# step 2 - iterate for a range of Ks and fit the pca components to the algorit
for k in range(1, 10):
    kmeans = KMeans(n_clusters = k, max_iter = 1000, random_state = 1).fit(ban
    sse[k] = kmeans.inertia_     # Use inertia attribute from the clustering o

# step 3
plt.figure()

plt.plot(list(sse.keys()), list(sse.values()), 'bx-')

plt.xlabel("Number of cluster")

plt.ylabel("SSE")

plt.show()
```



Obsevations

- We can see from the plot that there is a **big drop at K=3.(elbow).** Consistently dropping from 3 to 9.
- We may choose any number of clusters from 3 to 9 better to **choose 3.** WGSS(within group sum squares) beyond are minimal.

In [83]:
```
kmeans = KMeans(n_clusters = 3, random_state = 1)

kmeans.fit(bank_scaled)

# Adding predicted labels to the original data and the scaled data
bank_copy['KMeans_Labels'] = kmeans.predict(bank_scaled)

bank['KMeans_Labels'] = kmeans.predict(bank_scaled)
```

Create the cluster profiles using the summary statistics and box plots for each label

In [84]:
```
bank['KMeans_Labels'].value_counts()
```

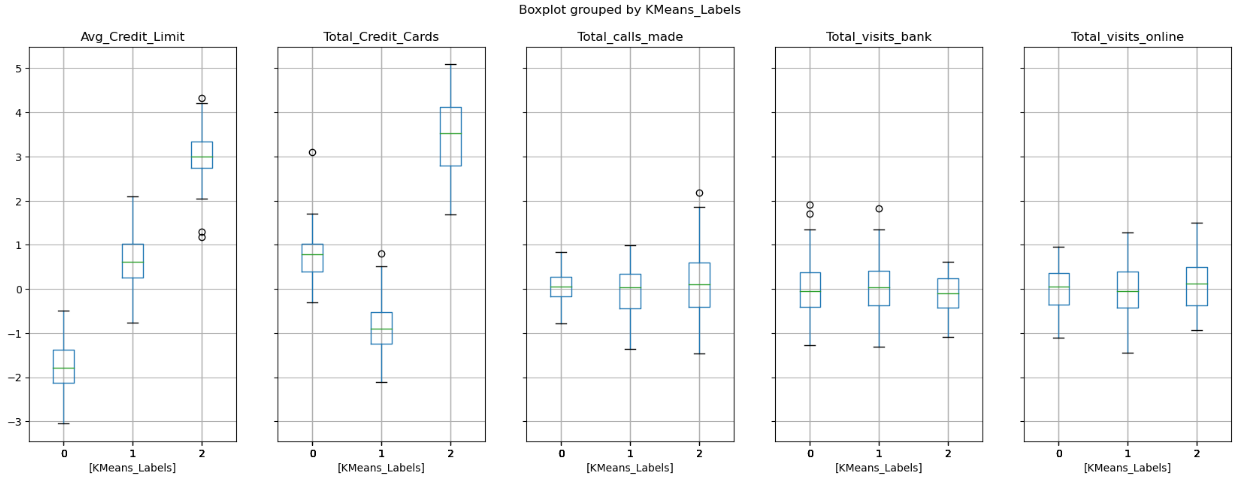
Out[84]:
```
1    378
0    221
2     50
Name: KMeans_Labels, dtype: int64
```

In [85]:
```
# Calculating summary statistics of the original data for each label
mean = bank.groupby('KMeans_Labels').mean()

median = bank.groupby('KMeans_Labels').median()

df_kmeans = pd.concat([mean, median], axis = 0)

df_kmeans.index = ['group_0 Mean', 'group_1 Mean', 'group_2 Mean', 'group_0 Med

df_kmeans.T
```

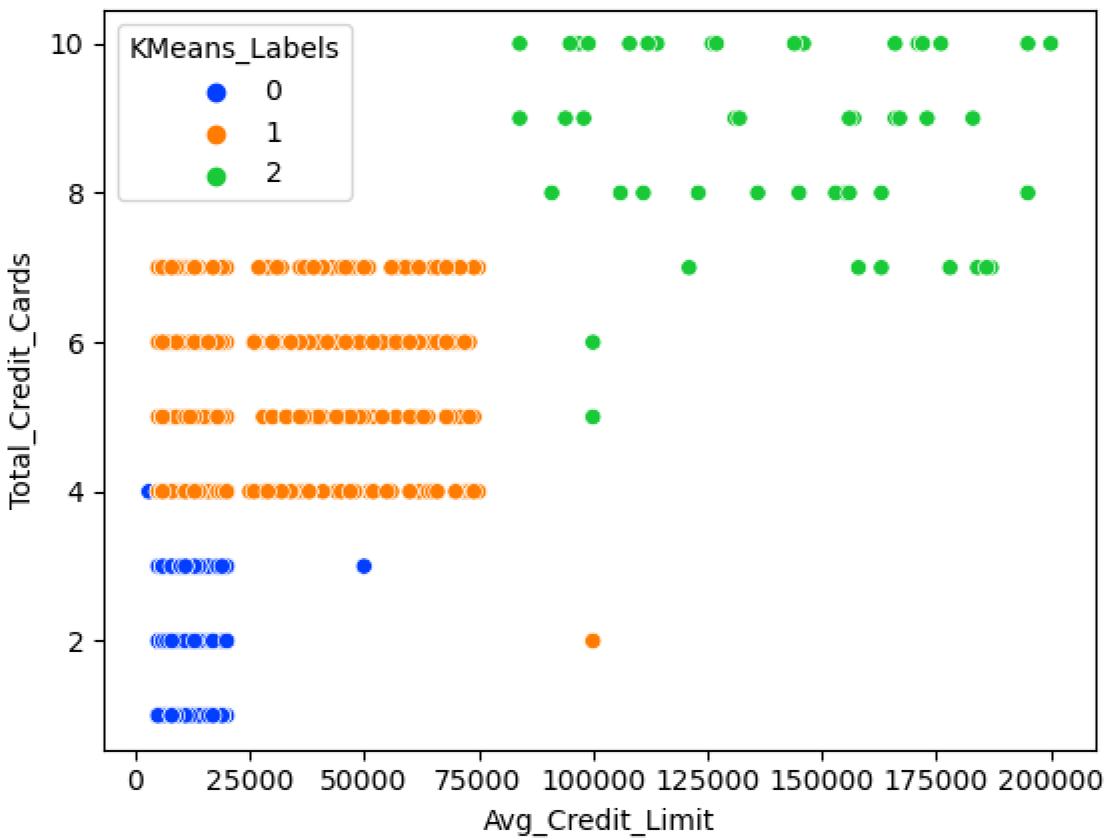
Out[85]:

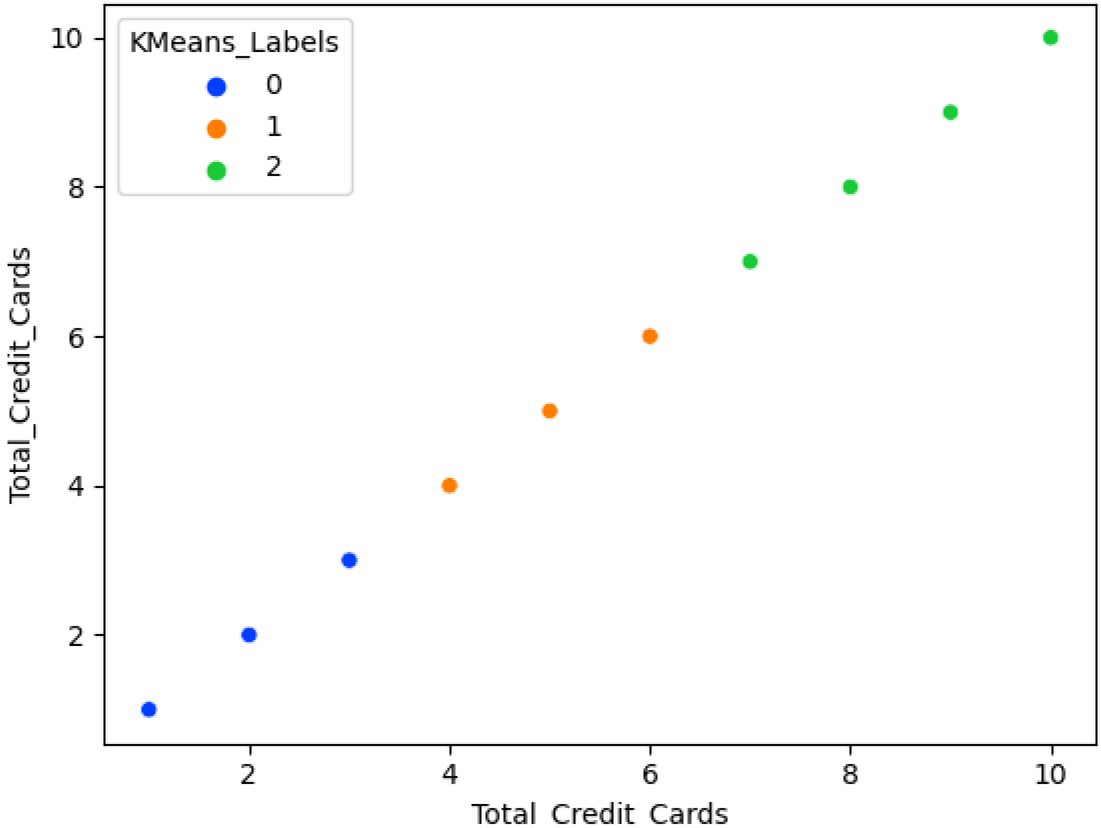
	group_0 Mean	group_1 Mean	group_2 Mean	group_0 Median	group_1 Median	group_2 Median
Avg_Credit_Limit	12239.819005	34071.428571	141040.00	12000.0	32000.0	145500.0
Total_Credit_Cards	2.411765	5.518519	8.74	2.0	6.0	9.0
Total_visits_bank	0.945701	3.484127	0.60	1.0	3.0	1.0
Total_visits_online	3.561086	0.981481	10.90	4.0	1.0	11.0
Total_calls_made	6.891403	1.992063	1.08	7.0	2.0	1.0

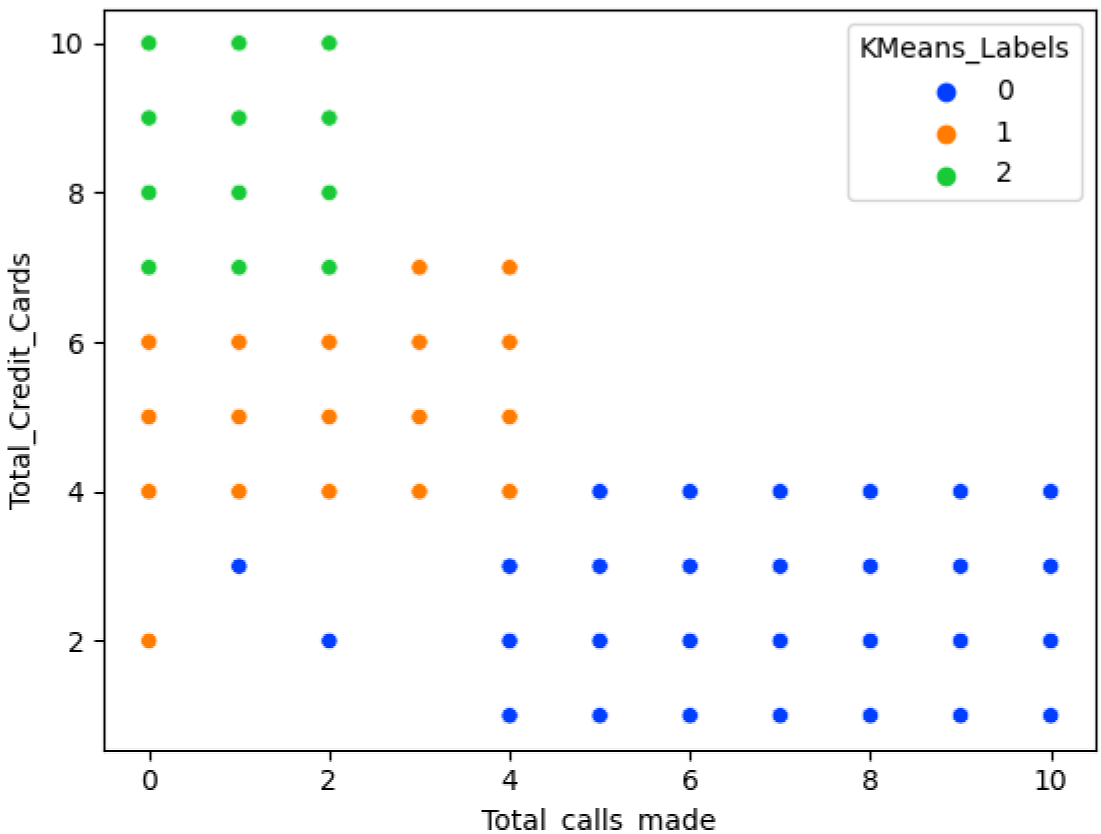
In [156…
```
# Visualizing different features of K-means.
bank_copy.boxplot(by = 'KMeans_Labels', layout = (1, 5), figsize = (10, 5))
plt.show()
```

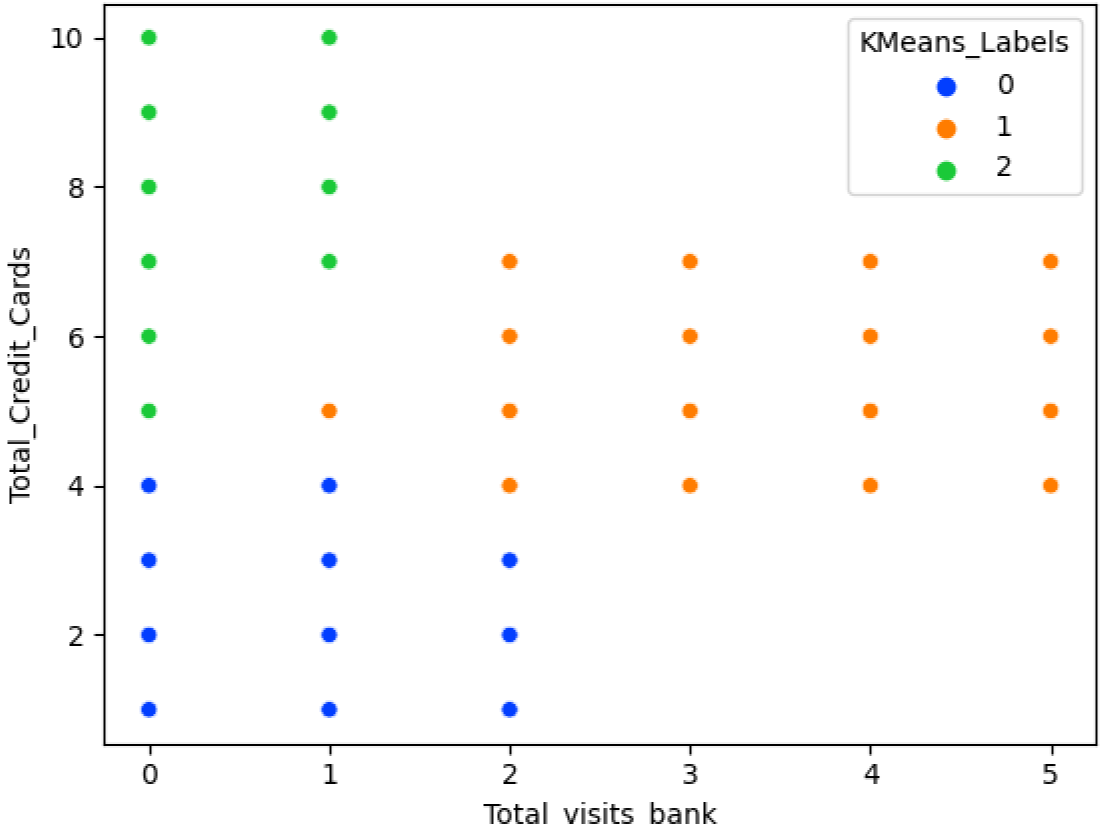


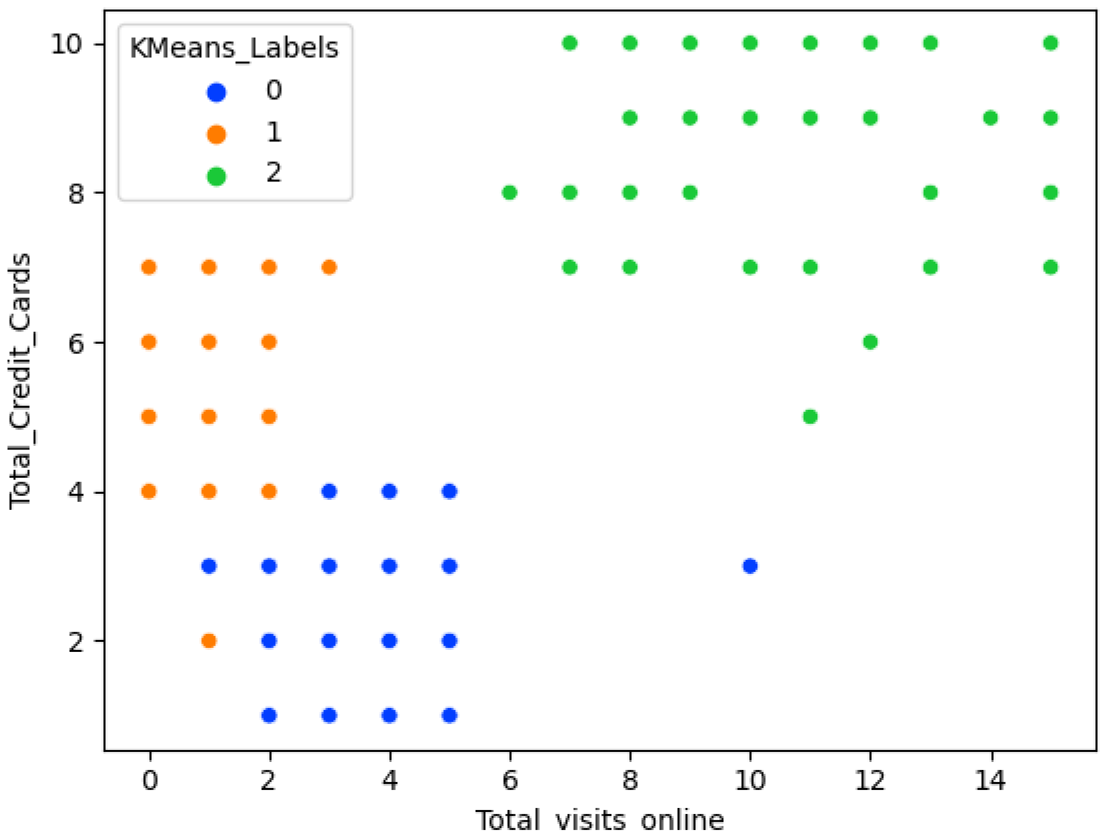
In [87]:
```
cols_visualise = ['Avg_Credit_Limit', 'Total_Credit_Cards',  'Total_calls_made

for col in cols_visualise:

    sns.scatterplot(x = col, y = 'Total_Credit_Cards', data = bank, hue = 'KMea

    plt.show()
```




KMeans_Labels
0
1
2
Total_Credit_Cards
Total_Credit_Cards
2
4
6
8
10


KMeans_Labels
0
1
2
Total_Credit_Cards
Total_calls_made
0
2
4
6
8
10





Cluster Profiles & Kmean labels:

Group 0:

- Has the lowest average and median credit limits, suggesting lower credit profile or income level.
- Average of approximately 2 credit cards.
- Has the lowest average and median number of credit cards, indicating lower credit usage or fewer credit accounts.
- Average of approximately 1 visits to the bank
- Average of approximately 4 visits to online.
- Average of approximately 7 calls.

Group 1:

- Average credit limit of approximately $35,000
- The average number of credit cards for individuals is 6.
- The average number of bank visits for individuals in grou 1 is 3.
- has the highest average and median number of bank visits, indicating a higher reliance on traditional banking services.
- The average number of online visits of 1.
- The average number of calls made by individuals is 2.

Group 3:

- Has the highest average and median credit limits, indicating that this group may consist of individuals with higher creditworthiness or higher incomes.
- Has the highest average and median number of credit cards, suggesting that these individuals may have more extensive credit usage or multiple credit accounts.
- has the lowest average of bank visits just 1.
- has the highest average and median number of online visits, with values of approximately 10.90 and 11, respectively.

Gaussian Mixture Model

Let's now create clusters using the Gaussian Mixture Model.

- Apply the Gaussian Mixture Model algorithm on the pca components

In [88]:
```
gmm = GaussianMixture(n_components = 3, random_state = 1)

gmm.fit(bank_scaled)

bank_copy['GmmLabels'] = gmm.predict(bank_scaled)

bank['GmmLabels'] = gmm.predict(bank_scaled)
```

Create the cluster profiles using the summary statistics and box plots for each label

In [89]:
```
bank.GmmLabels.value_counts()
```

Out[89]:
```
1    378
0    221
2     50
Name: GmmLabels, dtype: int64
```

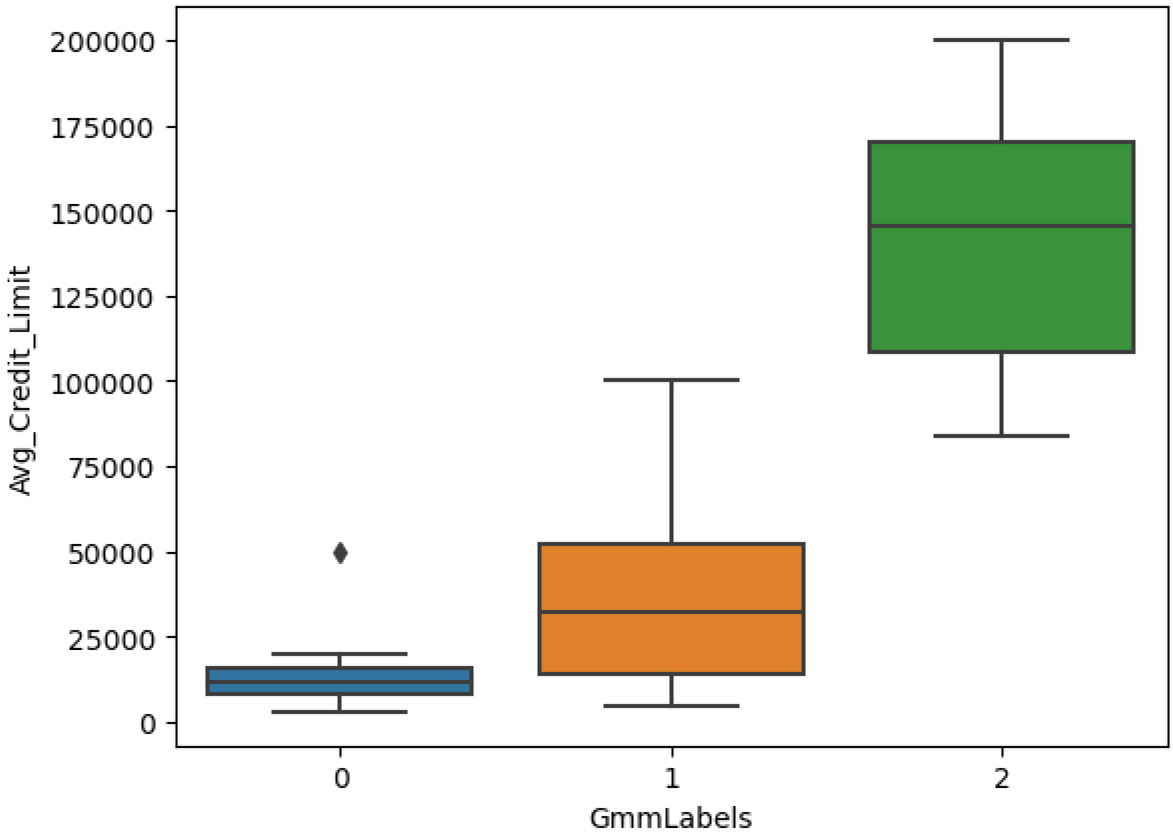
Compare the clusters from both algorithms - K-means and Gaussian Mixture Model

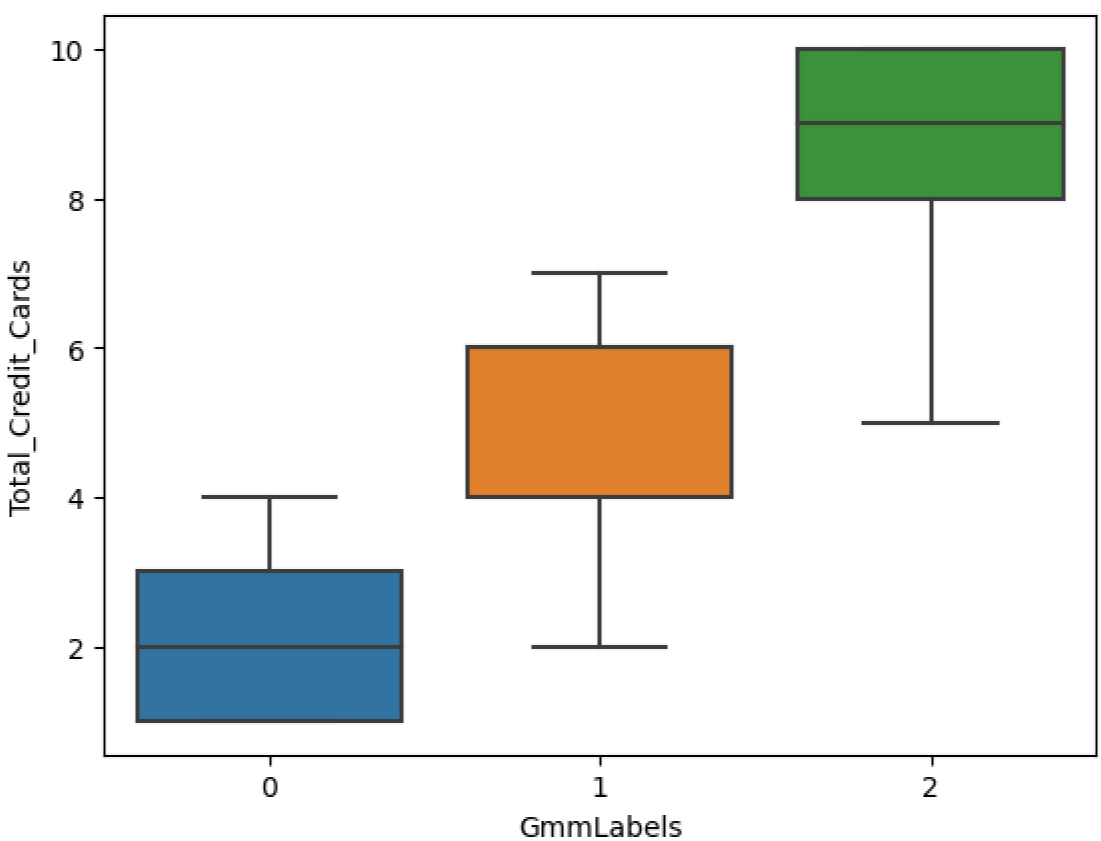
In [90]:
```
original_features = ['Avg_Credit_Limit','Total_Credit_Cards','Total_calls_made

mean = bank.groupby('GmmLabels').mean()

median = bank.groupby('GmmLabels').median()

df_gmm = pd.concat([mean, median], axis = 0)

df_gmm.index = ['group_0 Mean', 'group_1 Mean', 'group_2 Mean', 'group_0 Mediar

df_gmm[original_features].T
```

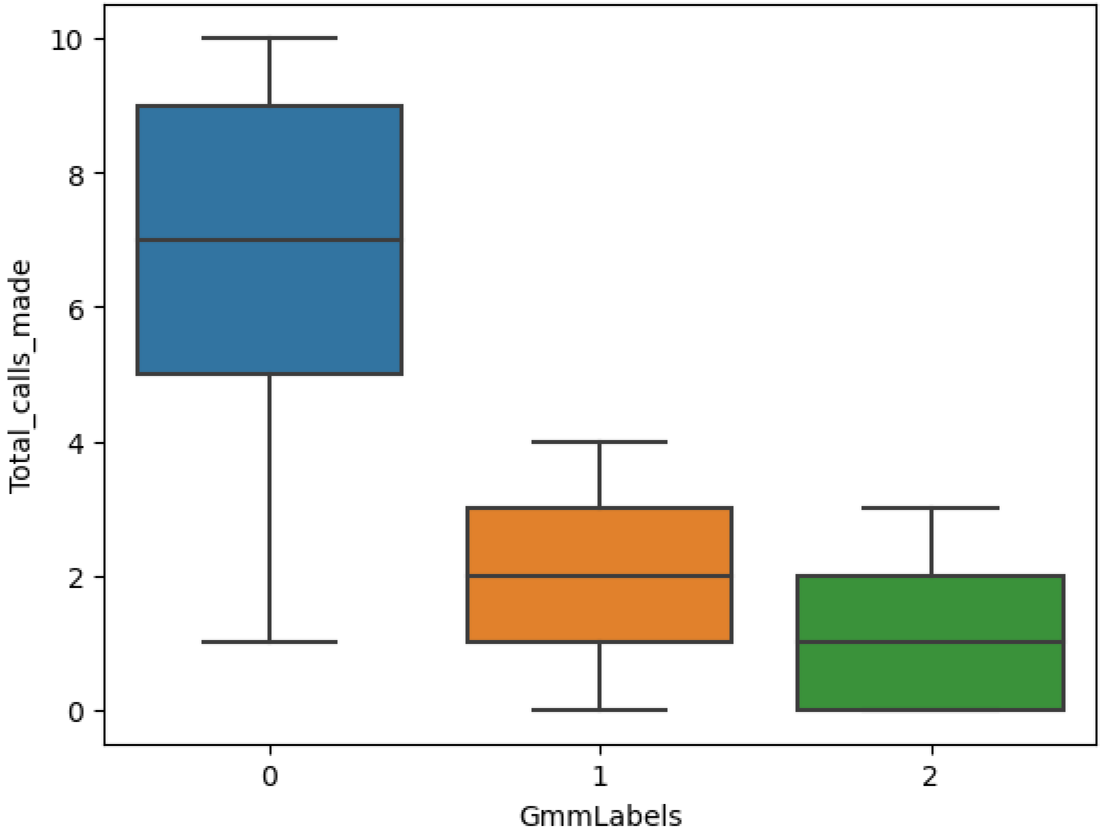
Out[90]:

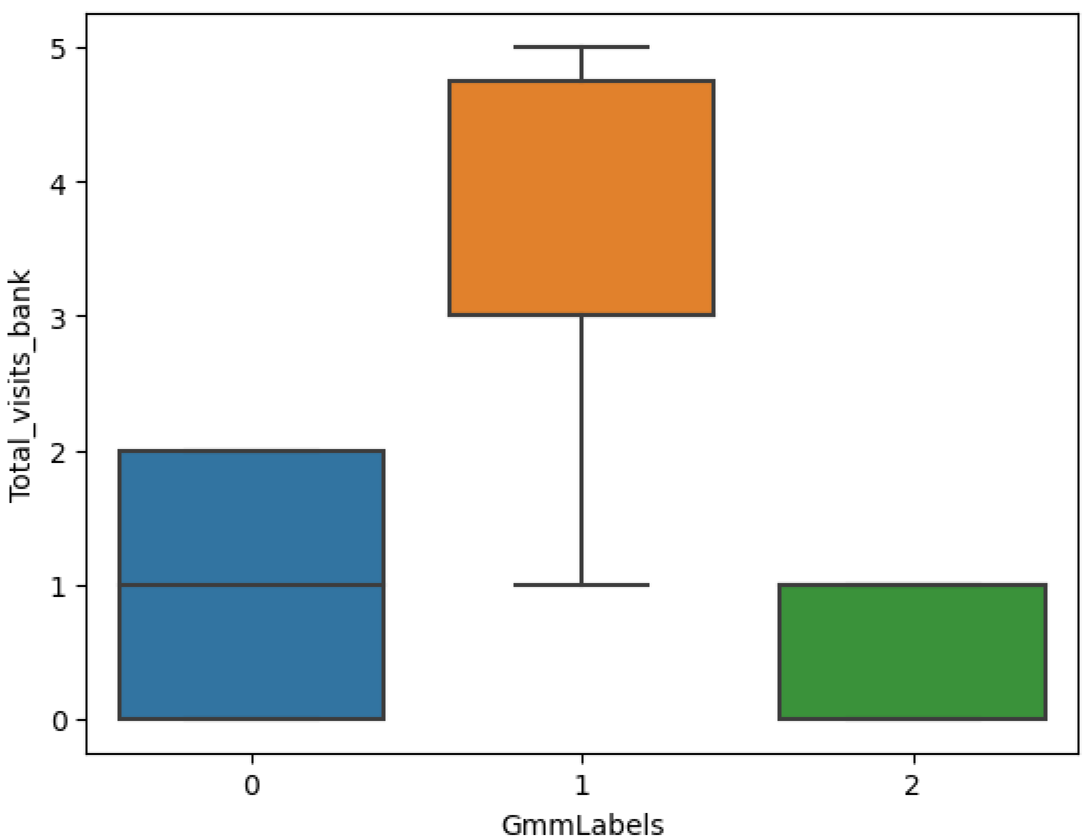
	group_0 Mean	group_1 Mean	group_2 Mean	group_0 Median	group_1 Median	group_2 Median
Avg_Credit_Limit	12239.819005	34071.428571	141040.00	12000.0	32000.0	145500.0
Total_Credit_Cards	2.411765	5.518519	8.74	2.0	6.0	9.0
Total_calls_made	6.891403	1.992063	1.08	7.0	2.0	1.0
Total_visits_bank	0.945701	3.484127	0.60	1.0	3.0	1.0
Total_visits_online	3.561086	0.981481	10.90	4.0	1.0	11.0

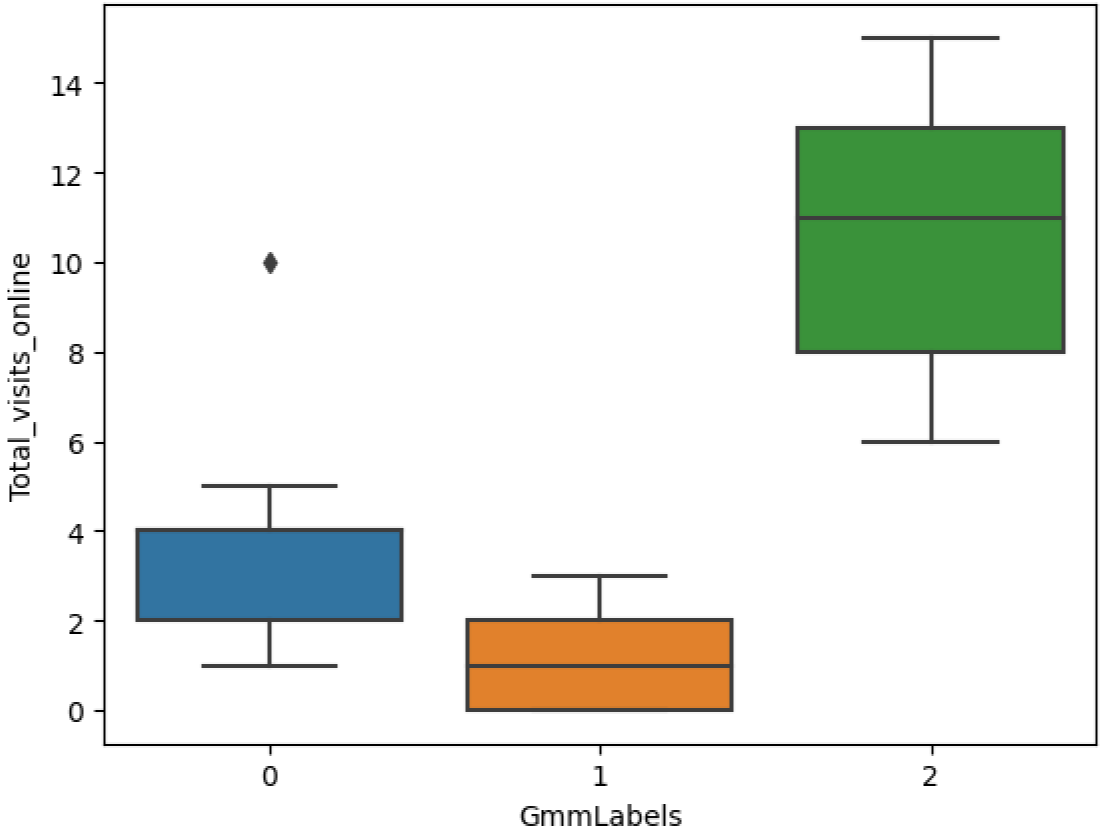
In [161…
```
cols_visualise = ['Avg_Credit_Limit','Total_Credit_Cards','Total_calls_made','

for col in cols_visualise:
    sns.boxplot(x = 'GmmLabels', y = col, data = bank)

    plt.show()
```


200000
175000
150000
125000
100000
75000
50000
25000
0
Avg_Credit_Limit
0
1
2
GmmLabels


10
8
6
4
2
Total_Credit_Cards
0
1
2
GmmLabels


Total_calls_made
10
8
6
4
2
0
0
1
2
GmmLabels


Total_visits_bank
5
4
3
2
1
0
0
1
2
GmmLabels



Observations: GMM Labels

Avg_Credit_Limit:

- The groups are divided based on credit limit
- with group 2 having the highest average credit limits,
- followed by group 1, and then group 0.

Total_Credit_Cards:

- The groups are distinguished by the number of credit cards,
- with group 2 having the highest average.

Total_calls_made:

- The groups are distinguished by the frequency of calls made by individuals,
- with group 0 having the highest average and median values,
- followed by group 1, and then group 2.

Total_visits_bank:

- The groups frequent the bank differently
- with group 1 having the highest average followed by group 0, and then group 2.

Total_visits_online:

- Group 2 has the highest average, indicating that these individuals are highly active in online banking activities.
- Group 0 represents individuals with a moderate level of online engagement,
- while group 1 has the lowest average.

K-Medoids

- Apply the K-Medoids clustering algorithm on the pca components

In [101…
```
kmedo = KMedoids(n_clusters = 3, random_state = 1)

kmedo.fit(bank_scaled) # imput data scaled

bank_copy['kmedoLabels'] = kmedo.predict(bank_scaled)

bank['kmedoLabels'] = kmedo.predict(bank_scaled)
```

In [102…
```
bank.kmedoLabels.value_counts()
```

Out[102]:
```
2    273
0    221
1    155
Name: kmedoLabels, dtype: int64
```

In [103…
```
# Calculating the mean and the median of the original data for each label
original_features = ['Avg_Credit_Limit', 'Total_Credit_Cards',  'Total_calls_ma

mean = bank.groupby('kmedoLabels').mean()

median = bank.groupby('kmedoLabels').median()

df_kmedoids = pd.concat([mean, median], axis = 0)

df_kmedoids.index = ['group_0 Mean', 'group_1 Mean', 'group_2 Mean', 'group_0 M

df_kmedoids[original_features].T
```

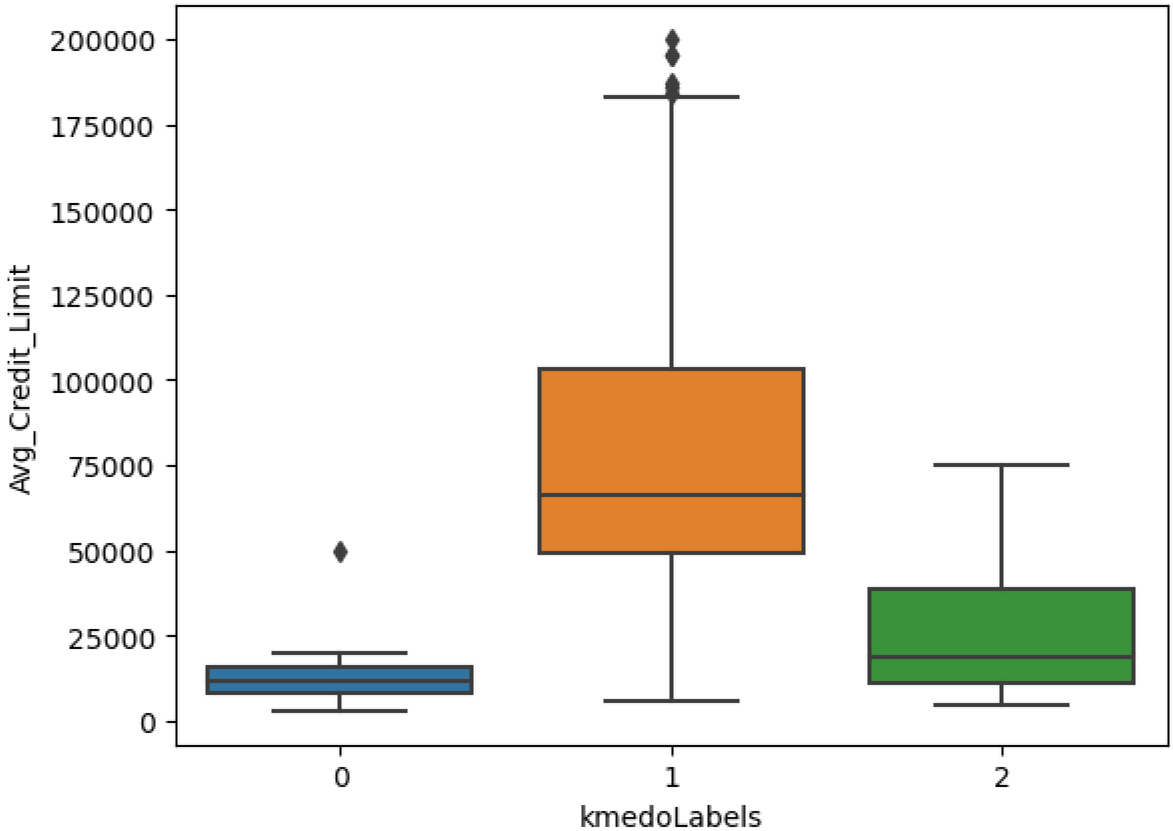
Out[103]:

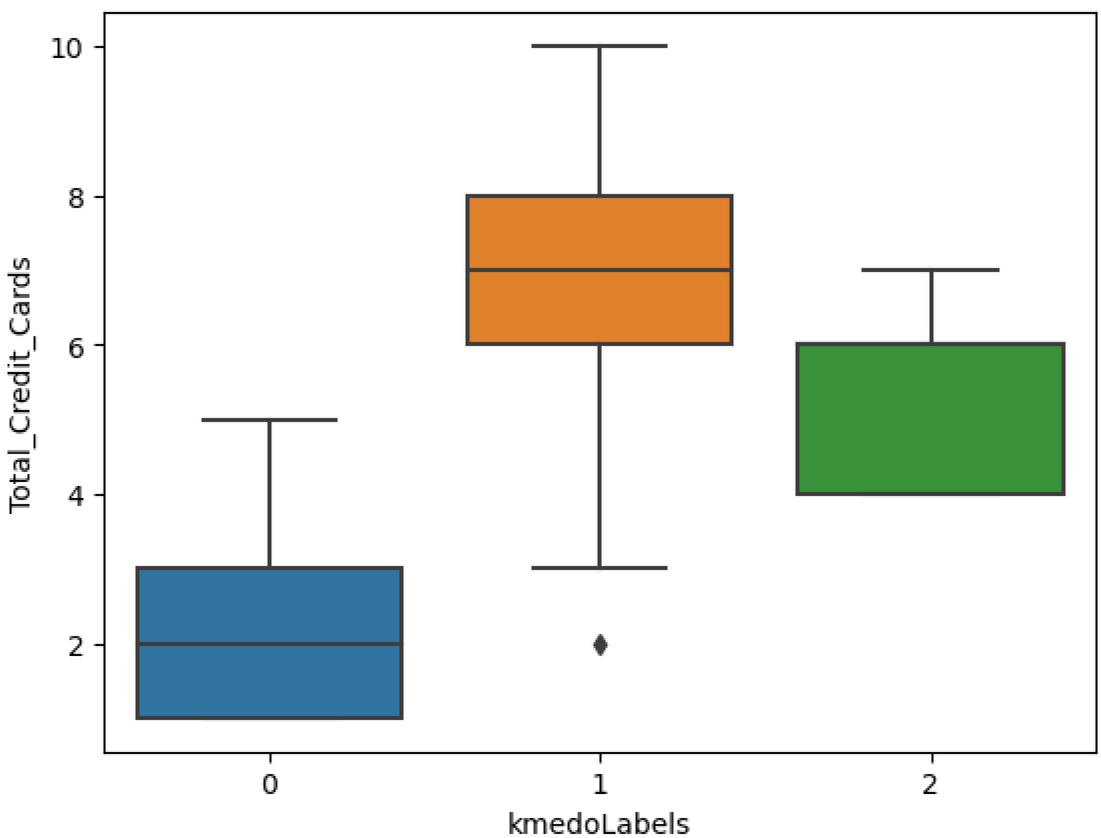
	group_0 Mean	group_1 Mean	group_2 Mean	group_0 Median	group_1 Median	group_2Med
Avg_Credit_Limit	12203.619910	80625.806452	27260.073260	12000.0	66000.0	190(
Total_Credit_Cards	2.420814	6.741935	5.406593	2.0	7.0	
Total_calls_made	6.904977	2.006452	1.805861	7.0	2.0	
Total_visits_bank	0.954751	1.800000	3.904762	1.0	2.0	
Total_visits_online	3.565611	4.187097	0.974359	4.0	2.0	

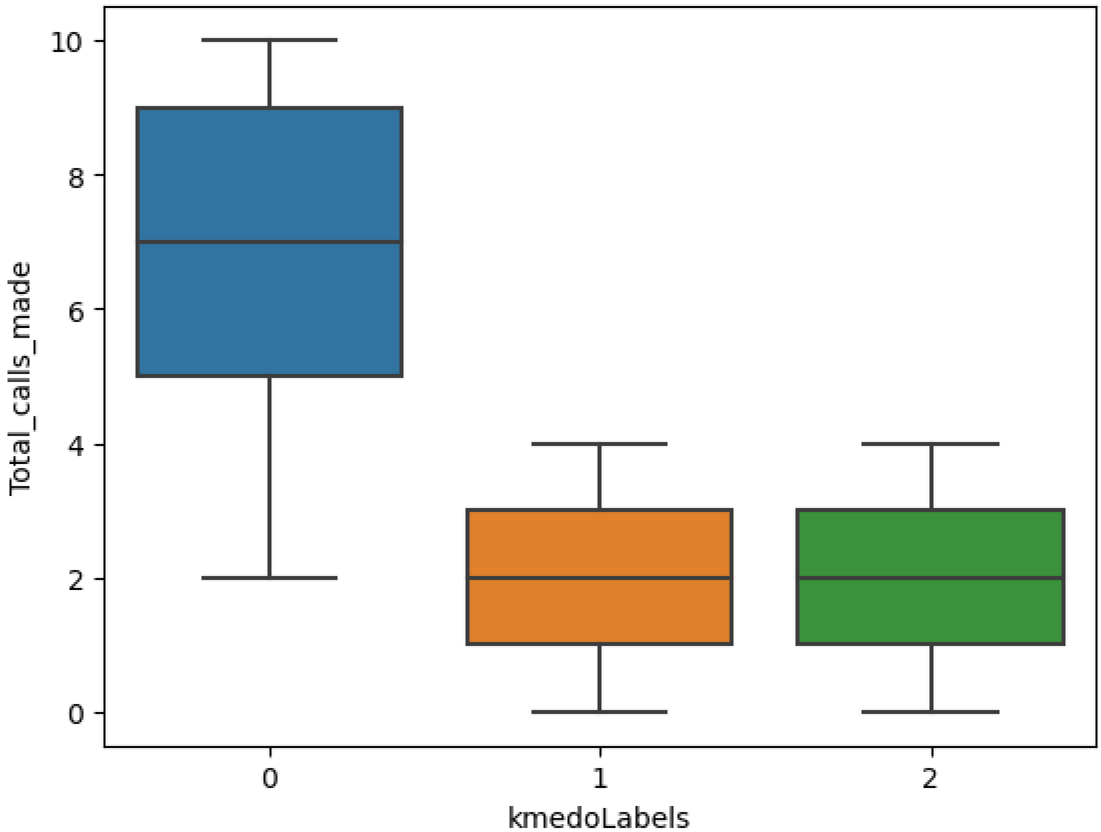
Create cluster profiles using the summary statistics and box plots for each label

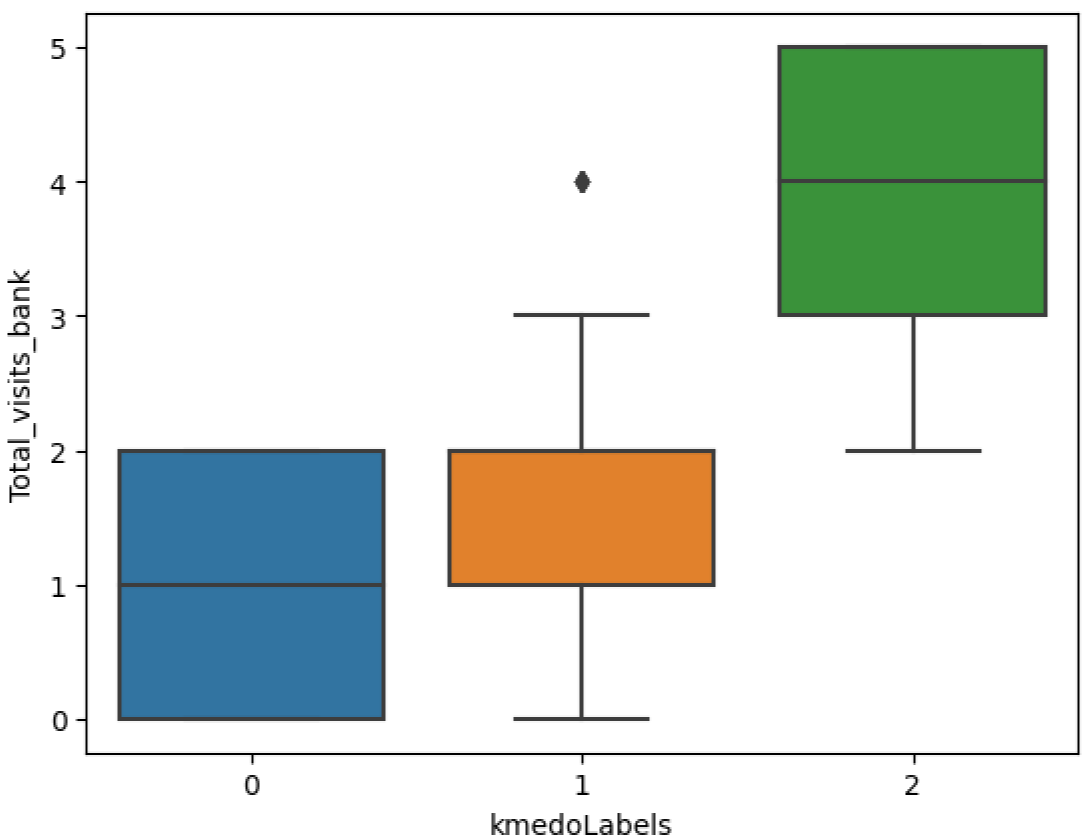
In [104…

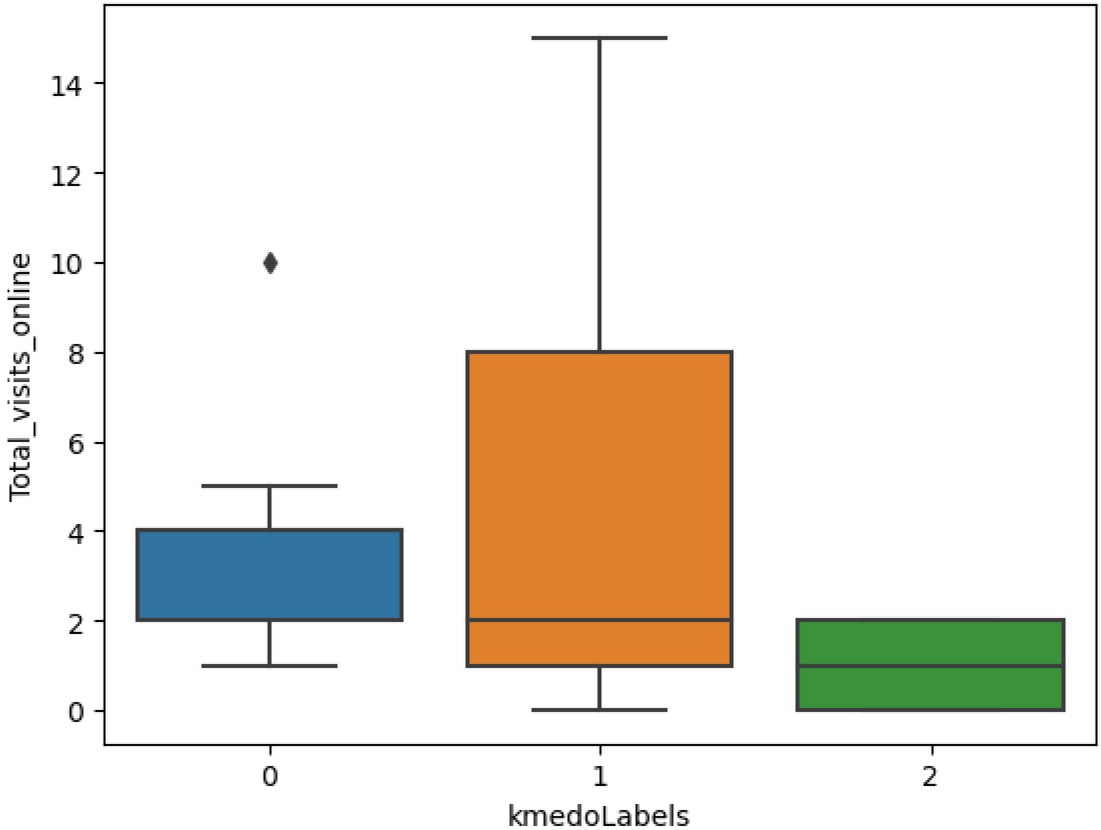
```
for col in cols_visualise:

    sns.boxplot(x = 'kmedoLabels', y = col, data = bank)

    plt.show()
```






10
8
6
4
2
0
Total_calls_made
0
1
2
kmedoLabels


5
4
3
2
1
0
Total_visits_bank
0
1
2
kmedoLabels



Cluster Profiles: KMedoids Labels

Consistent with the KMeans and Gmm labels in this case the groups break as followed:

Group 0:

- Individuals in group 0 have an average of approximately 2 credit cards and low credit line, and feel confortable making phone -calls to the bank.

Group 1:

- The average number of credit cards for individuals is approximately 7, good credit score and a stout presence online.

Group 2:

- has an average of approximately 5 credit cards, with This group represents moderate credit score and makes an average of 2 trips to the bank.

Compare the clusters from K-Means and K-Medoids

In [109…

```
comparison = pd.concat([df_kmedoids, df_kmeans], axis = 1)[original_features]

comparison
```

Out[109]:

	Avg_Credit_Limit	Avg_Credit_Limit	Total_Credit_Cards	Total_Credit_Cards	To
group_0 Mean	12203.619910	12239.819005	2.420814	2.411765	
group_1 Mean	80625.806452	34071.428571	6.741935	5.518519	
group_2 Mean	27260.073260	141040.000000	5.406593	8.740000	
group_0 Median	12000.000000	12000.000000	2.000000	2.000000	
group_1 Median	66000.000000	32000.000000	7.000000	6.000000	
group_2Median	19000.000000	NaN	6.000000	NaN	
group_2 Median	NaN	145500.000000	NaN	9.000000	

Comparing Clusters:

K-means and K-Medoids

- The comparison is consistant throught the data with one notable disparity.
- The **Average Credit Limit are inconsistant in group 1 there is a considerable difference
- of 45,000.**
- Group 2 also has an inconsistant difference of 12,000
- but it is closer
- and **consistant with the overall analysis.**

Conclusions and Business Recommendations

Objective and Goal:

According to the data there are 3 main focused groups with different characteristics. By catering to their preferences with a personalized campaing AllLife bank can target new customers and upsell to existing ones.

These are the following findings:

Group A:

- This group has the lowest average credit limits, suggesting a **lower credit profile or income level.**
- On average, individuals in this group have approximately 2 credit cards, indicating lower credit usage or fewer credit accounts.
- They make an average of approximately 1 visit to the bank, suggesting limited reliance on traditional banking services.
- Additionally, individuals in this **group make the most calls.**

Group B:

- Individuals in this group have an *average credit limit* of approximately $35,000.
- On average, they possess 5 or more credit cards, indicating a **moderate level of credit usage.**
- They make an average of 3 visits to the bank, suggesting a **higher reliance on traditional banking services.**
- Moderate online visits, individuals in this group there is moderate preference for online banking.
- The average number of calls made by individuals in this group is 2.

Group C:

- This group has the highest average and median credit limits.
- Indicating that its members may possess **higher creditworthiness or incomes.**
- Individuals in this group have the highest average and median number of credit cards, suggesting **extensive credit usage or multiple credit accounts.**
- They have the lowest average number of bank visits, indicating a lower reliance on traditional banking services, with an average of just 1 visit.
- On the other hand, they have the highest average and median number of online visits.
- This indicates a **strong preference for online banking services.**