In [2]:
```
#importing necessary libraries
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering

from scipy.stats import zscore
#Reading DataSet
data = pd.read_excel('Credit Card Customer Data.xlsx')
data.dtypes
```

Out[2]:
```
Sl_No                  int64
Customer Key           int64
Avg_Credit_Limit       int64
Total_Credit_Cards     int64
Total_visits_bank      int64
Total_visits_online    int64
Total_calls_made       int64
dtype: object
```

In [3]:
```
data.shape
```

Out[3]:
```
(660, 7)
```

In [4]:
```
data.head()
```

Out[4]:

	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_onlir
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	

In [5]:
```
data.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
```

- No null values, all are integers

In [6]:
```
data.nunique() #check unique values
```

Out[6]:
```
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
```

In [7]:
```
data_an = data.copy()
```

In [8]:
```
data_an.drop(['Sl_No','Customer Key'], axis=1,inplace=True)
```

In [9]:
```
data_an.head()
```

Out[9]:

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_mad
0	100000	2	1	1	
1	50000	3	0	10	
2	50000	7	1	3	
3	30000	5	1	1	
4	100000	6	0	12	

In [10]:
```
data_an.describe() #state observations on this
```

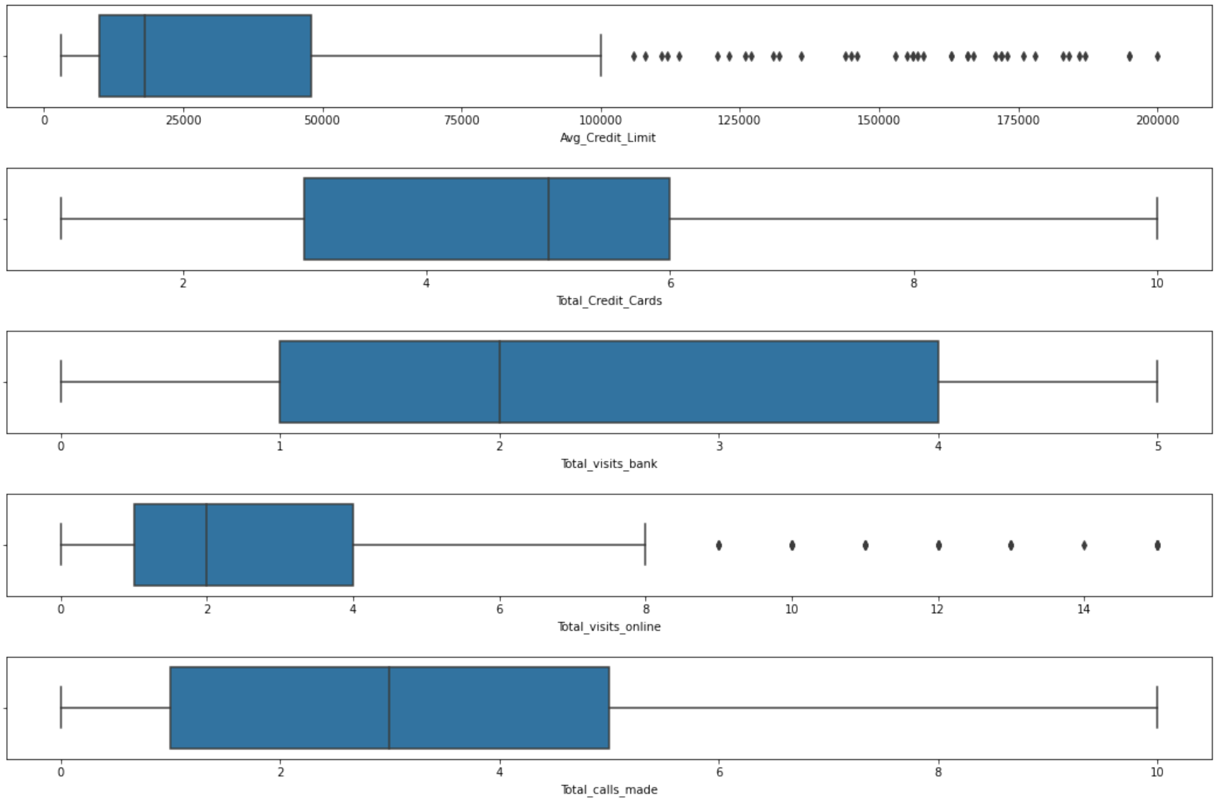
Out[10]:

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_
count	660.000000	660.000000	660.000000	660.000000	660.0(
mean	34574.242424	4.706061	2.403030	2.606061	3.5ℇ
std	37625.487804	2.167835	1.631813	2.935724	2.8
min	3000.000000	1.000000	0.000000	0.000000	0.0(
25%	10000.000000	3.000000	1.000000	1.000000	1.0(
50%	18000.000000	5.000000	2.000000	2.000000	3.0(
75%	48000.000000	6.000000	4.000000	4.000000	5.0(
max	200000.000000	10.000000	5.000000	15.000000	10.0(

- Positive Skewness shows on avg credit limit
- Online Visits + Calls made both have outliers

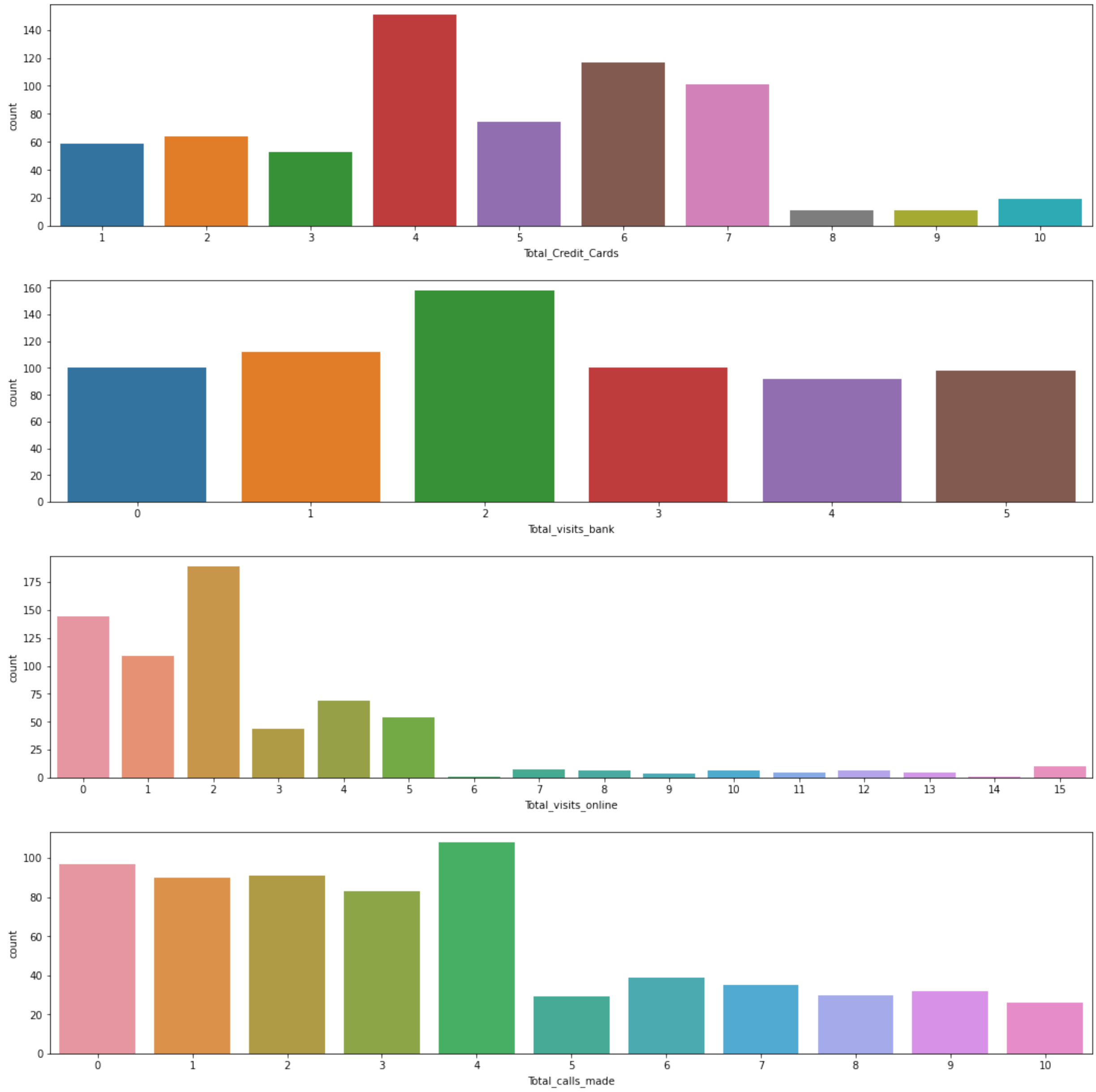
EDA

In [56]:
```
#univariate
data_columns = data_an.iloc[:,:].columns
fig, axes = plt.subplots(nrows=5, figsize=(15,10)) #creates seperate plots
counter = 0
for ii in range(5):
    sns.boxplot(ax=axes[ii], x=data_an[data_columns[counter]])
    counter = counter+1
fig.tight_layout(pad=2.0)
```



- Most customers have credit limit between 10,000 and 48,000. Customers with higher than average credit limit start from about 10,500 an up to 200,000, where as outliers, there is a significant number of customers in this range representing the top percentage of the database (will not remove)
- The high volume of customers have between 3 to 6 credit cards. #### Customer Service:
  - For all 3 query types, mostly 1 to 4 visits/calls were made. Outliers for online visits are kept, because there is no huge jumps between them and they form a constant line representing the higher number of visits

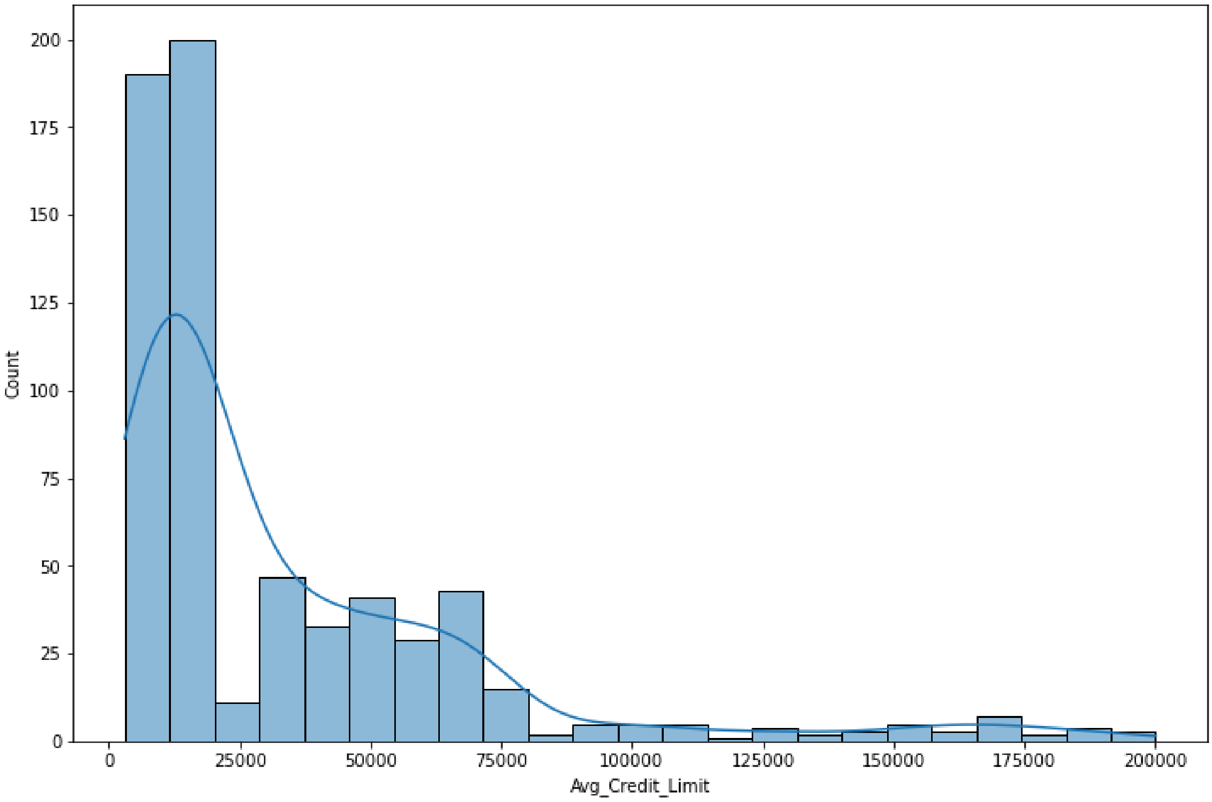
In [61]:
```
data_columns2 = data_an.iloc[:,1:].columns
fig, axes = plt.subplots(nrows=4, figsize=(15,15)) #creates seperate plots
counter = 0
for ii in range(4):
    sns.countplot(ax=axes[ii], x=data_an[data_columns2[counter]])
    counter = counter+1
fig.tight_layout(pad=2.0)
```



- Significantly, Online visits are mostly used across customers which is a good sign showing the bank shoul maintain their online system strategy
- Calls are also used frequenlty and suggests that it is a preferred service, meaning that the bank could implement an easier/faster call response in order to compete with the online service. i.e. a more simplified automated phone directory, to help shorten queue times or meet customer needs with only an automatic and secure telephone banking
- Maxmimum bank visits from the database is '5', other than showing that customers are relying more on phone or online queries, this could also mean that the bank may have only a small amount of local bank locations. If this is the case, the bank should work on implementing more accesible sub-branches.

In [79]:
```
fig, ax = plt.subplots(figsize=(12,8))
sns.histplot(x=data_an['Avg_Credit_Limit'], ax=ax, kde=True)
```

Out[79]: <AxesSubplot:xlabel='Avg_Credit_Limit', ylabel='Count'>

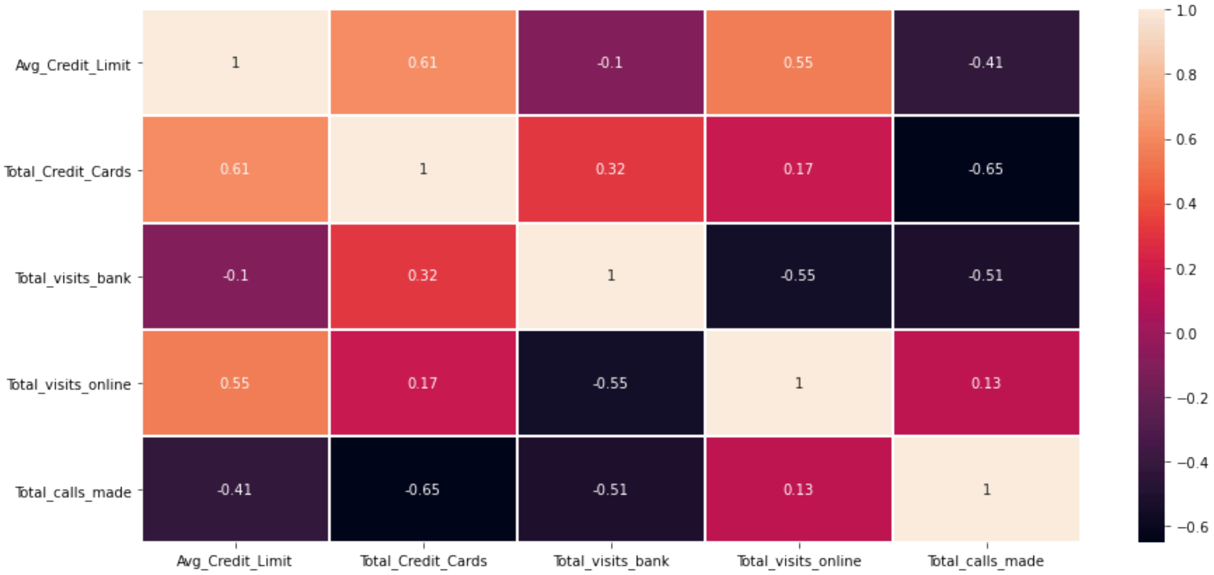


- The average credit limit for customers can act as a vital role to help maintain and bring in new customers.
- Seeing that there is a large number with credit limit between 5000 and 20000, this could be one type of credit card tier with basic necessary features. Also focusing on different credit card features and better marketing would help transfer the huge customer base from the lower credit limit to a higher limit.
- Then comes the second tier where a good number of customers have between 25000 and 75000, here there can be more loan options and higher discount capabilites with purchases + higher reward points
- Customers above the second tier would be considered the top 10 percent, the bank can have a special customer service offering credit features that adhere to the customer business needs.

Bivariate

In [80]:
```
plt.figure(figsize=(15,7))

sns.heatmap(data_an.corr(),annot=True,linewidth=1)
```

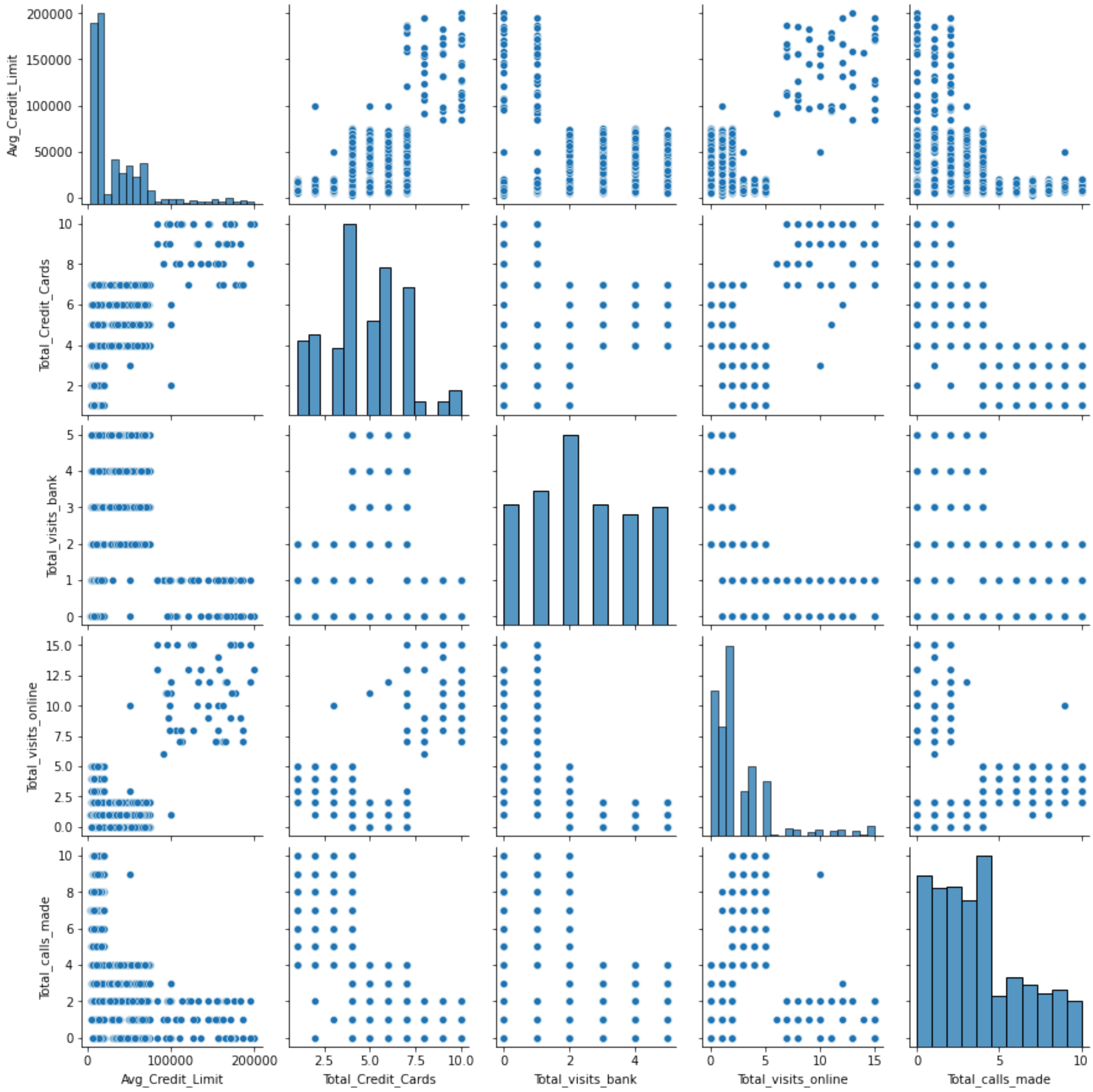
Out[80]: <AxesSubplot:>



- A very strong negative correlation between Total Calls Made and Total Credit Cards, suggests that customers have easy access to the online services and should focus on maintaining it's success

In [81]:
```
sns.pairplot(data_an)
```

Out[81]: <seaborn.axisgrid.PairGrid at 0x2628163ea60>



- Distribution clarifies that with more credit limit, customers tend to use the online service

K-Means Clustering:

In [23]:
```
# Scaling the data set before clustering
DataScaled = data_an.apply(zscore)
DataScaled.head(10)
```

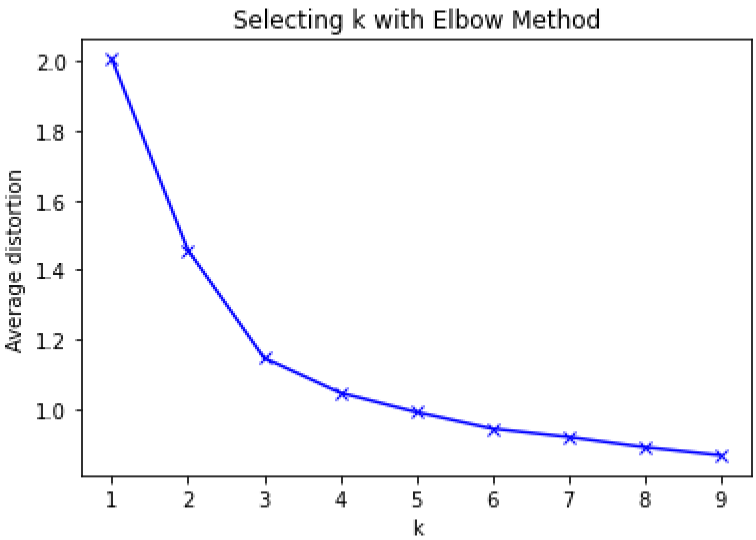
Out[23]:

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_mad
0	1.740187	-1.249225	-0.860451	-0.547490	-1.25153
1	0.410293	-0.787585	-1.473731	2.520519	1.89185
2	0.410293	1.058973	-0.860451	0.134290	0.14552
3	-0.121665	0.135694	-0.860451	-0.547490	0.14552
4	1.740187	0.597334	-1.473731	3.202298	-0.20373
5	-0.387644	-0.787585	-1.473731	-0.547490	1.54259
6	1.740187	0.135694	-1.473731	2.861408	-0.55300
7	-0.520633	-0.787585	-1.473731	-0.547490	-0.90227

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_mad
8	-0.786612	-1.249225	-1.473731	-0.206600	-0.55300
9	-0.839808	-0.325946	-1.473731	-0.547490	1.19332

In [84]:
```
from scipy.spatial.distance import cdist
clusters=range(1,10)
meanDistortions=[]

for k in clusters:
    model=KMeans(n_clusters=k)
    model.fit(DataScaled)
    prediction=model.predict(DataScaled)
    meanDistortions.append(sum(np.min(cdist(DataScaled, model.cluster_centers

plt.plot(clusters, meanDistortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Average distortion')
plt.title('Selecting k with Elbow Method')
```

Out[84]:
```
Text(0.5, 1.0, 'Selecting k with Elbow Method')
```



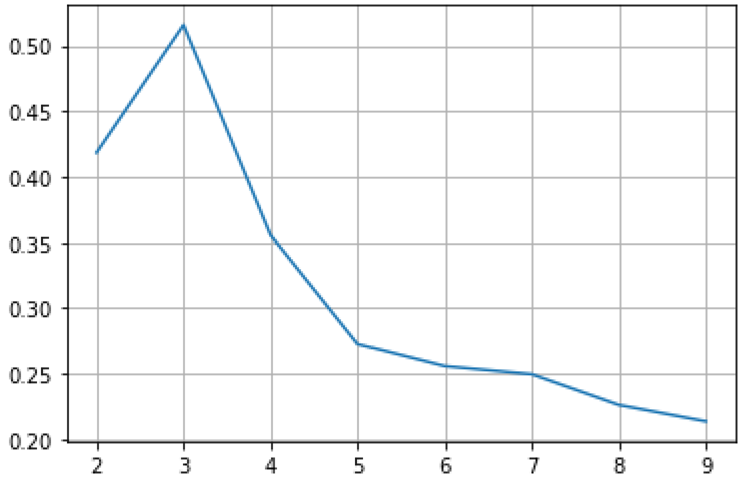
The Apropriate cluster number is 3 based on Elbow Method, lets check silhoutte scores:

In [25]:
```
from sklearn.metrics import silhouette_score
```

In [26]:
```
sil_score = []
cluster_list = list(range(2,10))
for n_clusters in cluster_list:
    clusterer = KMeans(n_clusters=n_clusters)
    preds = clusterer.fit_predict((DataScaled))
    score = silhouette_score(DataScaled, preds)
    sil_score.append(score)
    print("For n_clusters = {}, silhouette score is {})".format(n_clusters, s
```

```
For n_clusters = 2, silhouette score is 0.41842496663215445)
For n_clusters = 3, silhouette score is 0.5157182558881063)
For n_clusters = 4, silhouette score is 0.3556670619372605)
For n_clusters = 5, silhouette score is 0.2726898791817692)
For n_clusters = 6, silhouette score is 0.2558029066344975)
For n_clusters = 7, silhouette score is 0.24969750265579418)
For n_clusters = 8, silhouette score is 0.22627179769732209)
For n_clusters = 9, silhouette score is 0.21377817791261602)
```

In [27]:
```
plt.plot(cluster_list,sil_score)
plt.grid()
```



- silhouette scores prove that 3 is the optimal k number of clusters

KMeans cluster analysis:

In [85]:
```
final_model=KMeans(3)
final_model.fit(DataScaled)
```

Out[85]:
```
KMeans(n_clusters=3)
```

In [86]:
```
DataScaled["GROUP"] = final_model.labels_
```

In [88]:
```
cluster_profile = DataScaled.groupby("GROUP").mean()
```

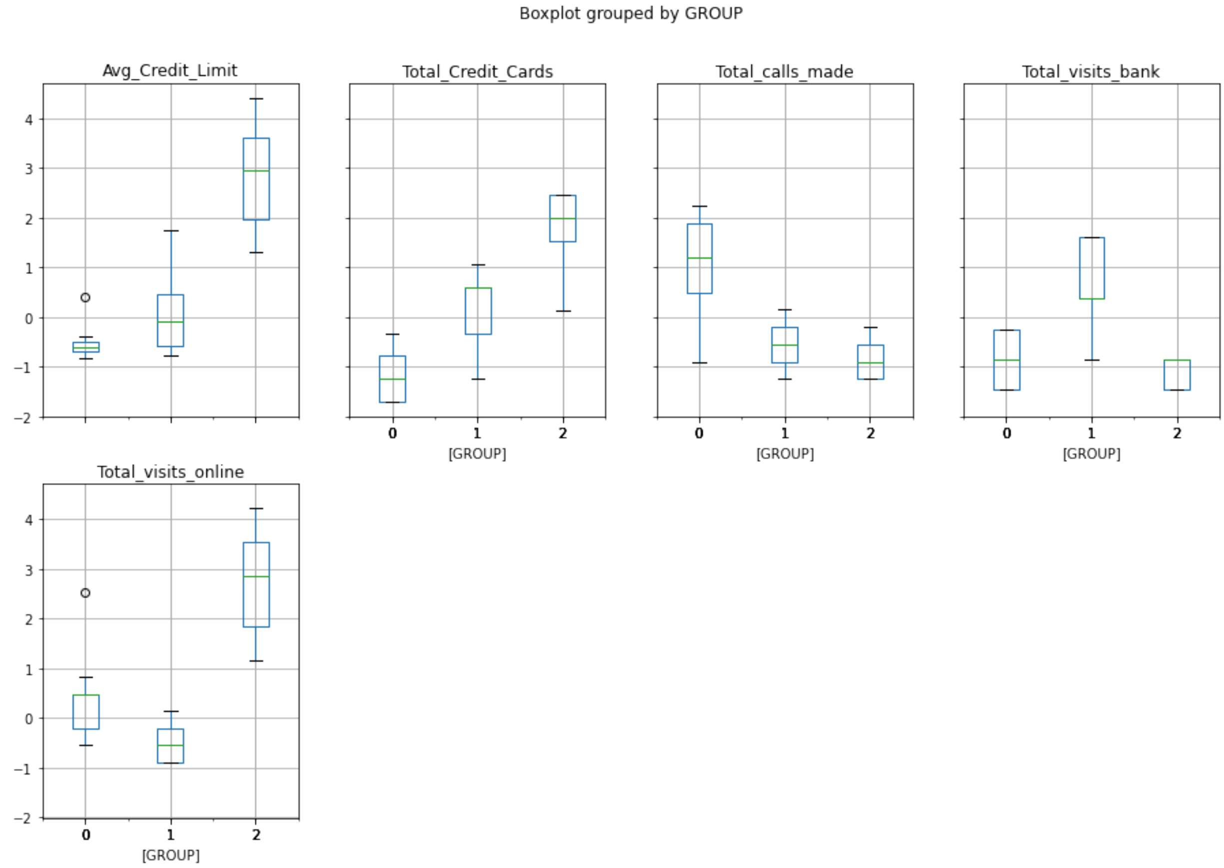
In [89]:
```
cluster_profile #give insight with box plots as well
```

Out[89]:

GROUP	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls
0	-0.595796	-1.059623	-0.901518	0.322997	1
1	-0.021062	0.373690	0.666395	-0.553672	-0.
2	2.831764	1.862226	-1.105763	2.827319	-0.

In [90]:
```
DataScaled.boxplot(by="GROUP", layout = (2,4),figsize=(15,10))
```

Out[90]:
```
array([[<AxesSubplot:title={'center':'Avg_Credit_Limit'}, xlabel='[GROUP]'>,
        <AxesSubplot:title={'center':'Total_Credit_Cards'}, xlabel='[GROU
P]'>,
        <AxesSubplot:title={'center':'Total_calls_made'}, xlabel='[GROUP]'>,
        <AxesSubplot:title={'center':'Total_visits_bank'}, xlabel='[GROU
P]'>],
       [<AxesSubplot:title={'center':'Total_visits_online'}, xlabel='[GROU
P]'>,
        <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



Insights

```
- cluster 0:
    - Average Credit Limit is very low
    - Low number of credit cards
    - very low number of bank visits
    - medium number of online visits
    - high number of calls made
- cluster 1:
    - Average Credit Limit is medium
    - medium number of credit cards
    - high number of bank visits
    - very low number of online visits
    - low number of calls made
- cluster 2:
    - Average Credit Limit is very high
    - high number of credit cards
    - very low number of bank visits
    - very high number of online visits
    - very low number of calls made
```

Heirarchal Clustering:

In [91]:
```
DataScaled2 = DataScaled.copy() #creating a copy of original scaled data
```

In [94]:
```
DataScaled2.drop(['GROUP'], axis=1,inplace=True) #'GROUP' was created from KM
```

In [95]:
```
DataScaled2.head()
```

Out[95]:

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_mad
0	1.740187	-1.249225	-0.860451	-0.547490	-1.25153
1	0.410293	-0.787585	-1.473731	2.520519	1.89185
2	0.410293	1.058973	-0.860451	0.134290	0.14552
3	-0.121665	0.135694	-0.860451	-0.547490	0.14552
4	1.740187	0.597334	-1.473731	3.202298	-0.20373

In [97]:

```
from scipy.spatial.distance import pdist
from scipy.cluster.hierarchy import cophenet, dendrogram, linkage
```

In [101…

```
#comparing different linkage methods with cophentic correlations
# closer the cophenet is to 1, the better for clustetering
linkage_methods = ['single', 'complete', 'average', 'weighted','median','ward
high_cophenet_corr = 0
high_lm = [0] ##should only use lm***
for lm in linkage_methods:
    Z = linkage(DataScaled2, metric='euclidean',method=lm)
    c, coph_dists = cophenet(Z, pdist(DataScaled2))
    print('Cophenetic Index for linkage method {} is {}'.format(lm,c))
    if high_cophenet_corr < c:
        high_cophenet_corr = c
        high_lm[0] = lm
```

```
Cophenetic Index for linkage method single is 0.7391220243806552
Cophenetic Index for linkage method complete is 0.8599730607972423
Cophenetic Index for linkage method average is 0.8977080867389372
Cophenetic Index for linkage method weighted is 0.8861746814895477
Cophenetic Index for linkage method median is 0.8893799537016724
Cophenetic Index for linkage method ward is 0.7415156284827493
```

In [102…

```
print('Highest cophenet correlation is {}, which is obtinaed with {} linkage
```

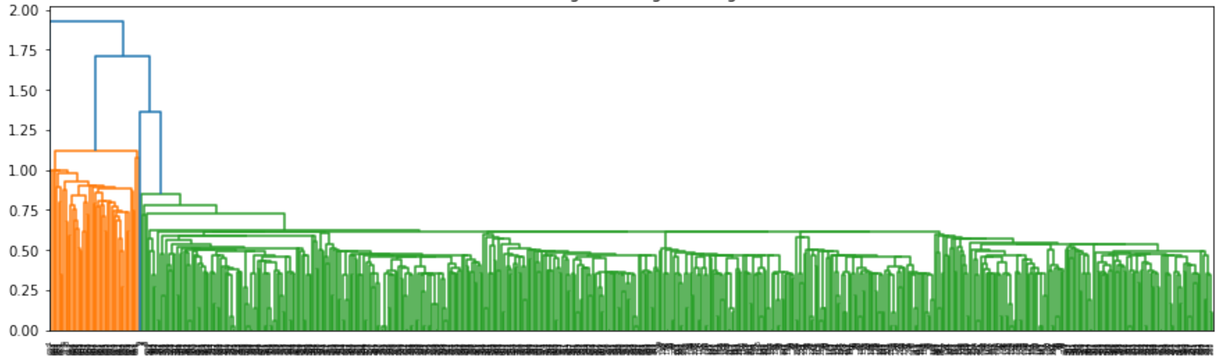
```
Highest cophenet correlation is 0.8977080867389372, which is obtinaed with av
erage linkage method
```

Dendrogram insight:

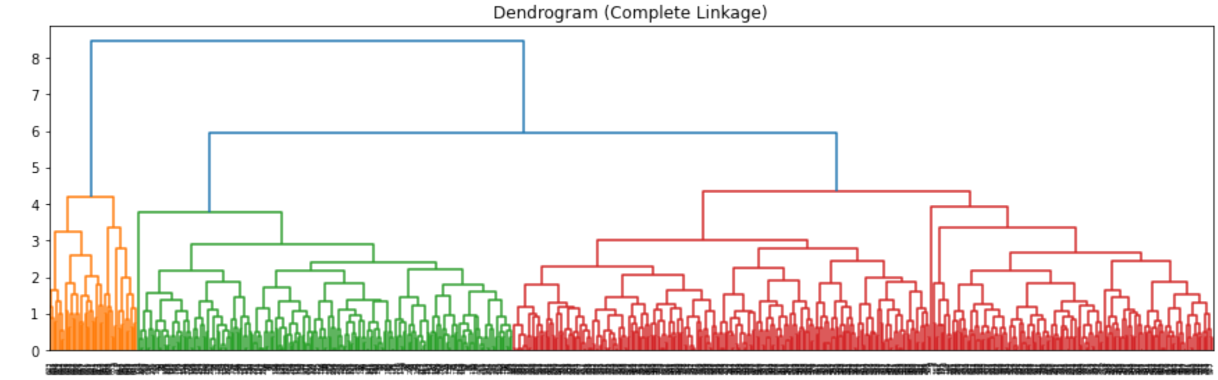
In [104…

```
methods = ['single', 'complete', 'average', 'weighted','median','ward']

# Create lists to save results of coph calculation
compare_cols = ['Linkage', 'Cophenetic Coefficient']
compare = []

#subplot image for each method
fig, axs = plt.subplots(len(methods), 1, figsize=(15,30))

for i, method in enumerate(methods):
    Z = linkage(DataScaled2, metric='euclidean', method=method)

    dendrogram(Z, ax=axs[i]);
    axs[i].set_title(f'Dendrogram ({method.capitalize()} Linkage)')
    coph_corr, coph_dist = cophenet(Z, pdist(DataScaled2))
    compare.append([method, coph_corr])
```

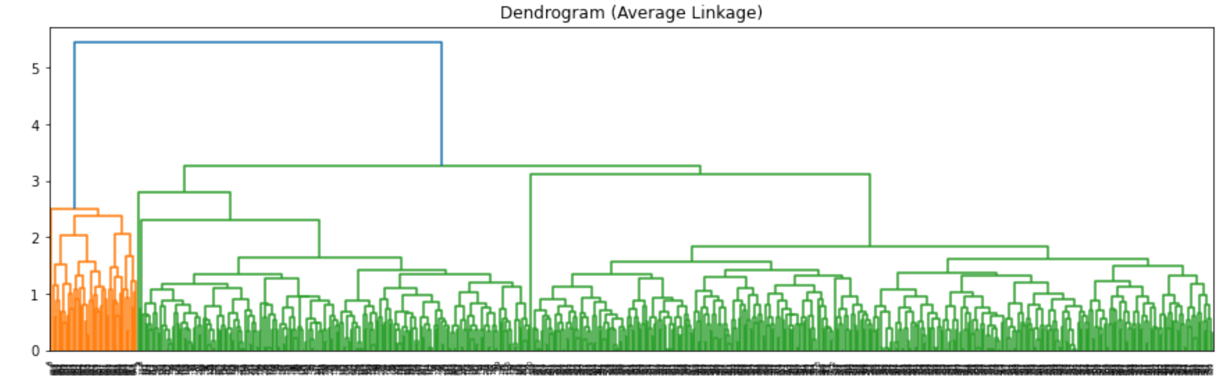
Dendrogram (Single Linkage)



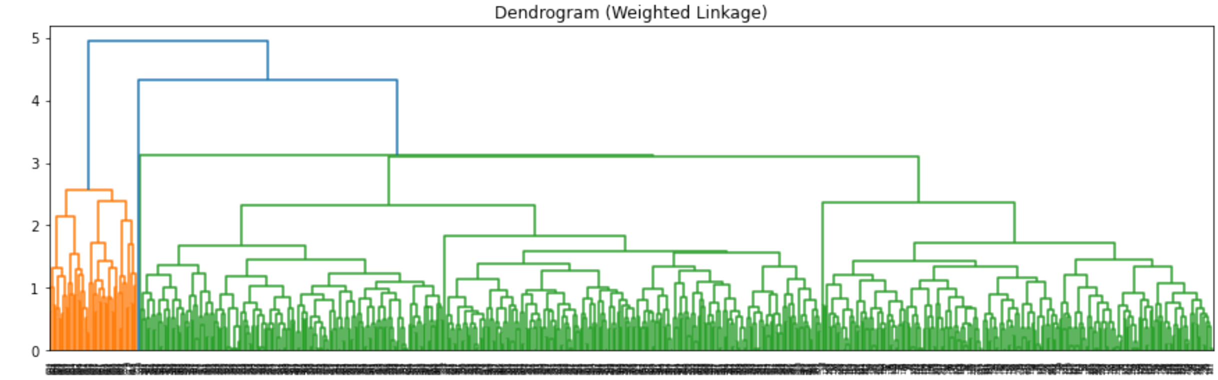
Dendrogram (Complete Linkage)



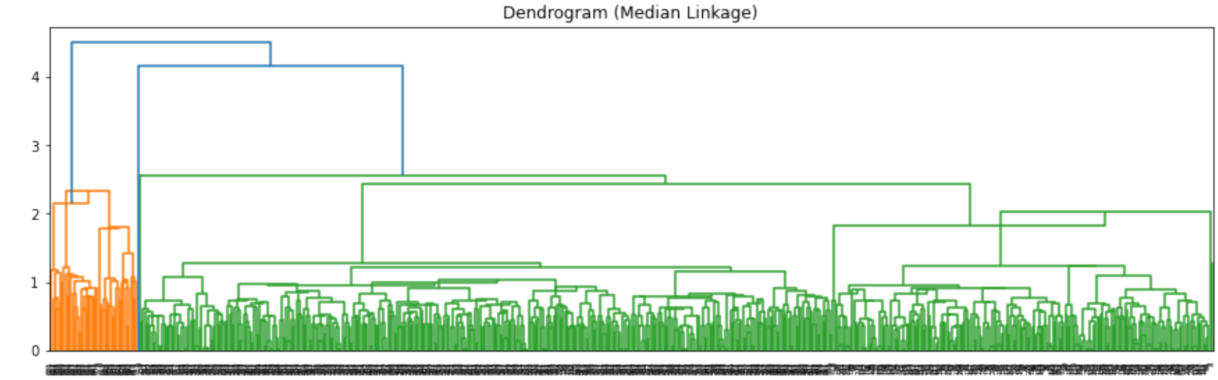
Dendrogram (Average Linkage)



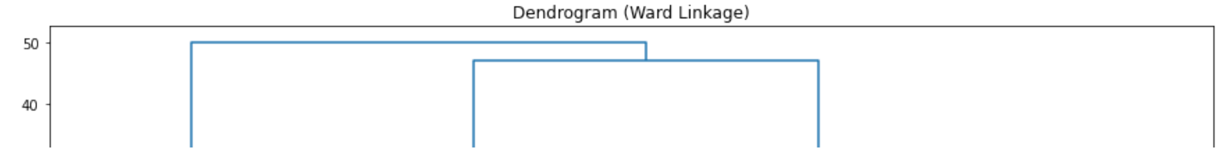
Dendrogram (Weighted Linkage)

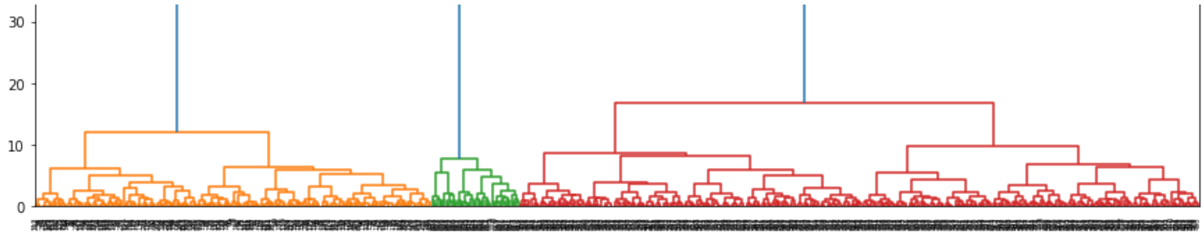


Dendrogram (Median Linkage)



Dendrogram (Ward Linkage)





Observation

- Dendogram with the ward linkage method two clear seperate clusters with distinct sub clusters

In [105…
```
compare_df = pd.DataFrame(compare, columns=compare_cols)
compare_df
```

Out[105…

	Linkage	Cophenetic Coefficient
0	single	0.739122
1	complete	0.859973
2	average	0.897708
3	weighted	0.886175
4	median	0.889380
5	ward	0.741516

- 3 clusters would be appropriate based on the dendogram with ward linkage method. Although other methods show higher cophenetic coefficients, they do not represen distinct clusters compared to 'Ward'

Ward 3 cluster build:

In [106…
```
Hmodel = AgglomerativeClustering(n_clusters=3,affinity='euclidean', linkage='
Hmodel.fit(DataScaled2)
DataScaled2['Heirarchal_Clusters'] = Hmodel.labels_
```

In [107…
```
H_cluster_profile = DataScaled2.groupby('Heirarchal_Clusters').mean()
```

In [108…
```
H_cluster_profile
```

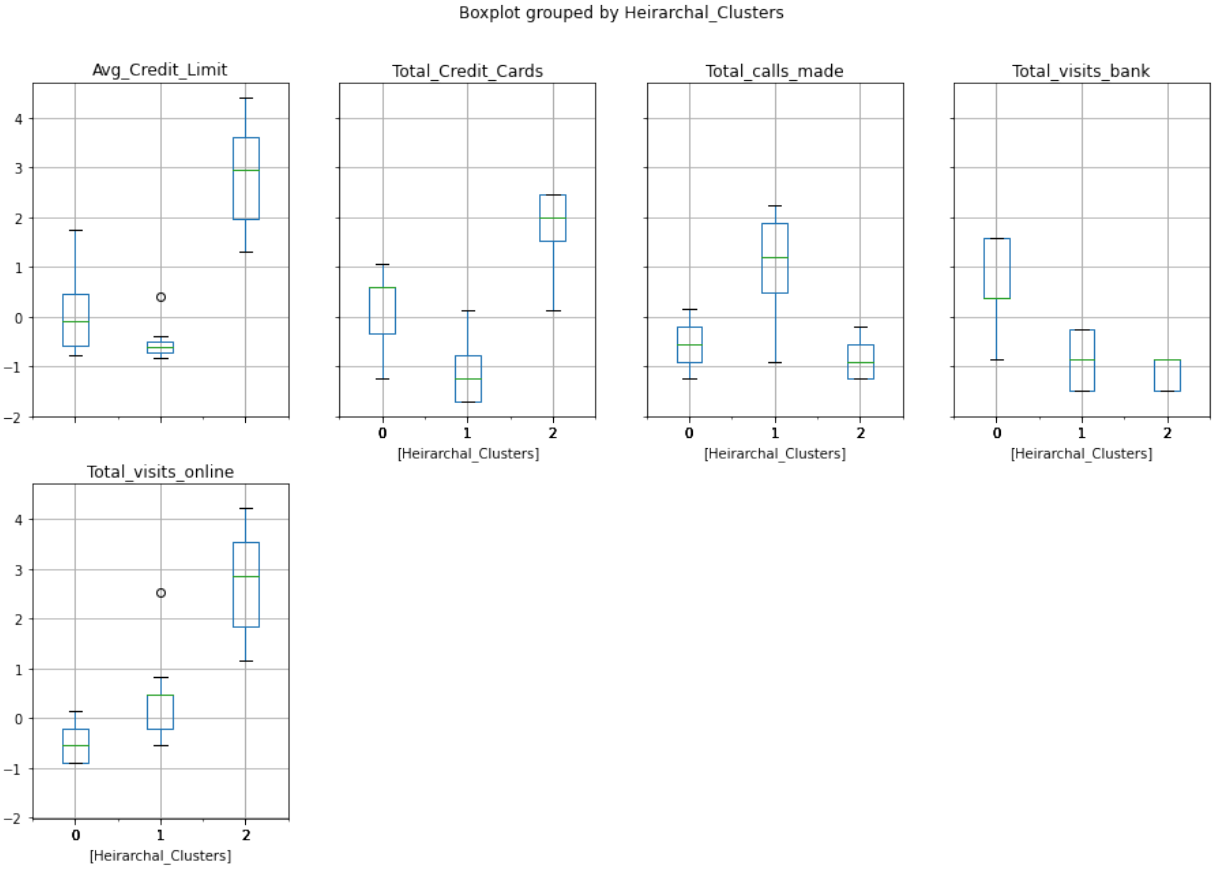
Out[108…

	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online
Heirarchal_Clusters				
0	-0.019212	0.374308	0.668767	-0.55457
1	-0.596408	-1.054310	-0.898610	0.32064
2	2.831764	1.862226	-1.105763	2.82731

In [109…
```
DataScaled2.boxplot(by="Heirarchal_Clusters", layout = (2,4),figsize=(15,10))
```

Out[109…
```
array([[<AxesSubplot:title={'center':'Avg_Credit_Limit'}, xlabel='[Heirarchal
_Clusters]'>,
        <AxesSubplot:title={'center':'Total_Credit_Cards'}, xlabel='[Heirarch
al_Clusters]'>,
        <AxesSubplot:title={'center':'Total_calls_made'}, xlabel='[Heirarchal
_Clusters]'>,
```

```
       <AxesSubplot:title={'center':'Total_visits_bank'}, xlabel='[Heirarcha
l_Clusters]'>],
       [<AxesSubplot:title={'center':'Total_visits_online'}, xlabel='[Heirarc
hal_Clusters]'>,
        <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



Heirarchal cluster Insights:

```
- cluster 0:
    - Average Credit Limit is low
    - medium number of credit cards
    - high number of bank visits
    - low number of online visits
    - low number of calls made
- cluster 1:
    - Average Credit Limit is low
    - low number of credit cards
    - very low number of bank visits
    - medium number of online visits
    - high number of calls made
- cluster 2:
    - Average Credit Limit is very high
    - high number of credit cards
    - very low number of bank visits
    - very high number of online visits
    - very low number of calls made

    ____________________________
```

Recalling KMeans Cluster Insights:

```
- cluster 0:
    - Average Credit Limit is very low
```

```
        - Low number of credit cards
        - very low number of bank visits
        - medium number of online visits
        - high number of calls made
    - cluster 1:
        - Average Credit Limit is medium
        - medium number of credit cards
        - high number of bank visits
        - very low number of online visits
        - low number of calls made
    - cluster 2:
        - Average Credit Limit is very high
        - high number of credit cards
        - very low number of bank visits
        - very high number of online visits
        - very low number of calls made
```

- Both methods showed similar grouping, where cluster '2' shows the customers targeted as higher tier credit users, whereas a high focus on online queries shows a robust online service model that can also be built on to attract tier 0 and tier 1 customers.
- Other than Average Credit card limit as shown in EDA being a key factor in clustering, the number of credit cards is seen to be increased as the credit limit increases. An example to have a better insight for the bankers is to expect higher bank visits from customers with a few number of credit cards.