In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

In [2]:
```
df = pd.read_csv('Credit Card Customer Data.csv')
df.head(5)
```

Out[2]:

	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
0	1	87073	100000	2	1	1	0
1	2	38414	50000	3	0	10	9
2	3	17341	50000	7	1	3	4
3	4	40496	30000	5	1	1	4
4	5	47437	100000	6	0	12	3

In [3]:
```
df.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 [4]:
```
df.describe()
```

Out[4]:

	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
count	660.000000	660.000000	660.000000	660.000000	660.000000	660.000000	660.000000
mean	330.500000	55141.443939	34574.242424	4.706061	2.403030	2.606061	3.583333
std	190.669872	25627.772200	37625.487804	2.167835	1.631813	2.935724	2.865317
min	1.000000	11265.000000	3000.000000	1.000000	0.000000	0.000000	0.000000
25%	165.750000	33825.250000	10000.000000	3.000000	1.000000	1.000000	1.000000
50%	330.500000	53874.500000	18000.000000	5.000000	2.000000	2.000000	3.000000
75%	495.250000	77202.500000	48000.000000	6.000000	4.000000	4.000000	5.000000
max	660.000000	99843.000000	200000.000000	10.000000	5.000000	15.000000	10.000000

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Sl_No                  0
Customer Key           0
Avg_Credit_Limit       0
Total_Credit_Cards     0
Total_visits_bank      0
Total_visits_online    0
Total_calls_made       0
dtype: int64
```

In [6]:
```
df.duplicated().sum()
```

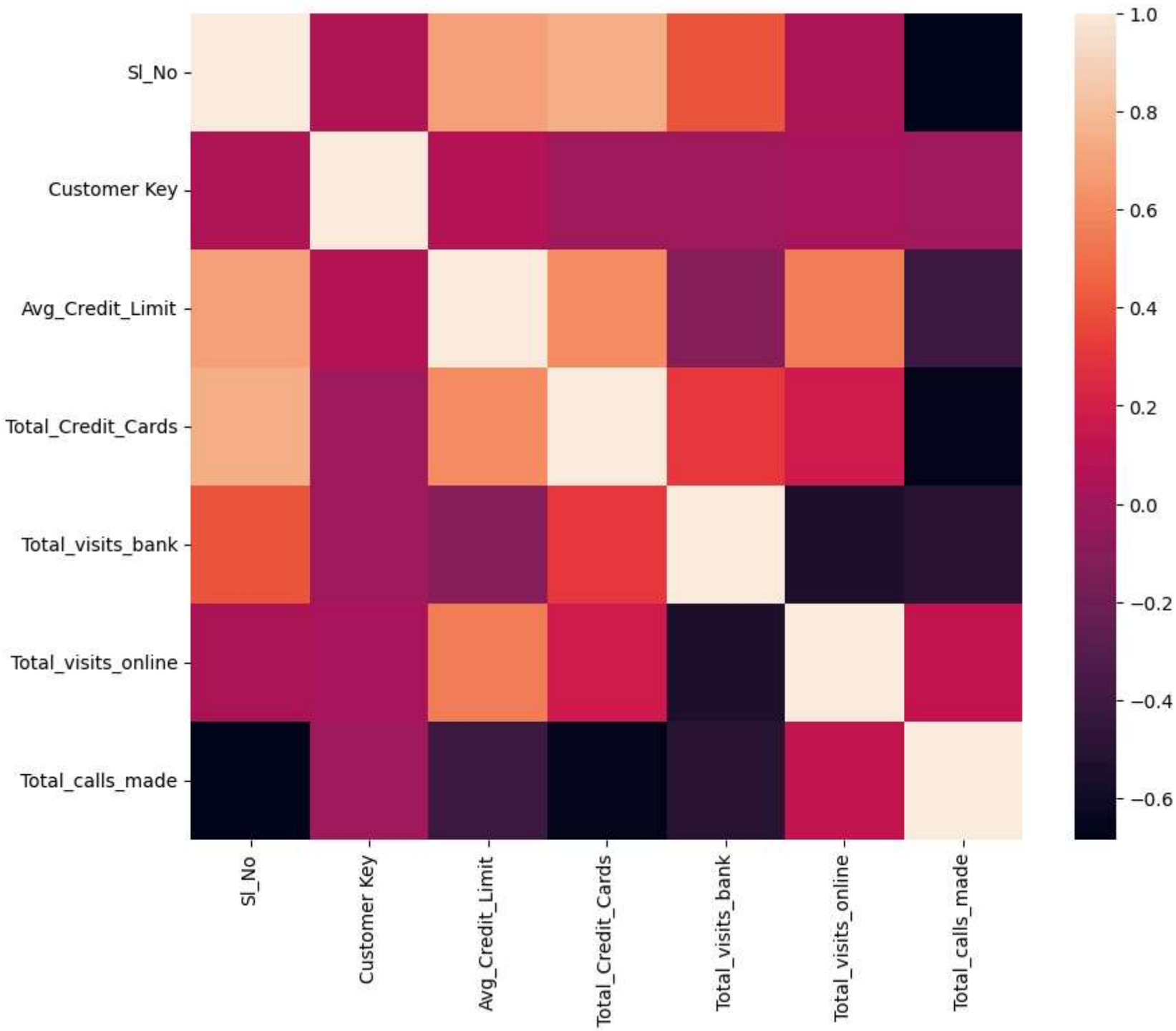
Out[6]:
```
0
```

In [7]:
```
corr_matrix = df.corr(numeric_only = True)
corr_matrix
```

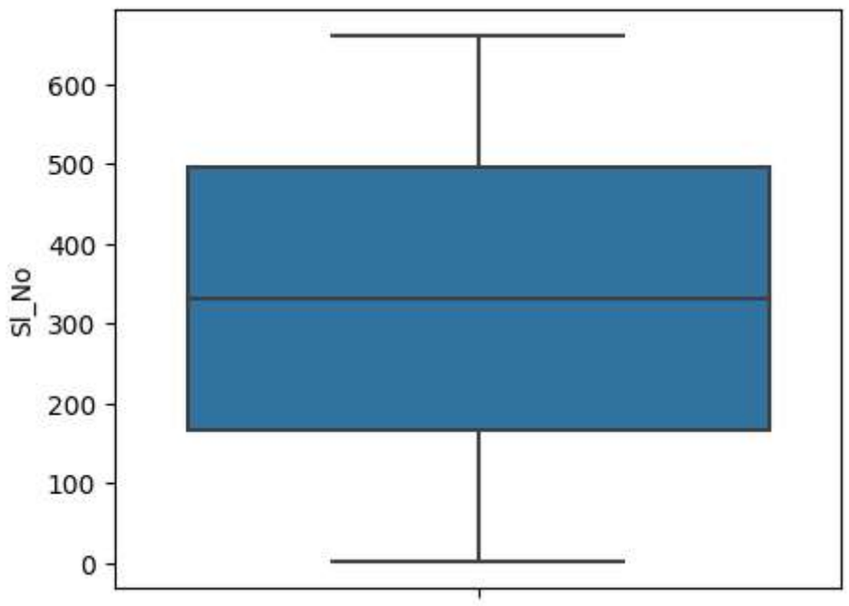
Out[7]:

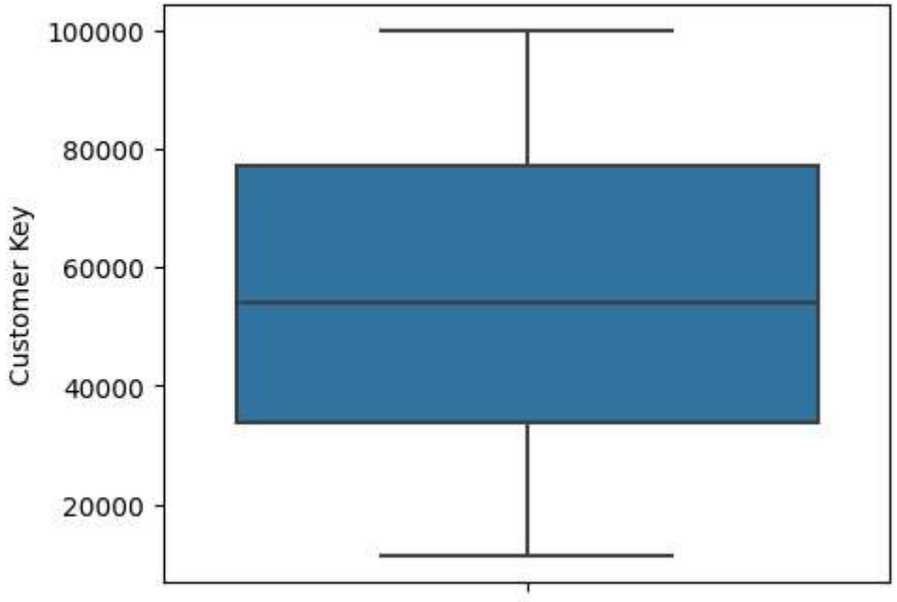
	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
Sl_No	1.000000	0.052886	0.677962	0.739329	0.406438	0.033916	-0.684125
Customer Key	0.052886	1.000000	0.068604	-0.010281	-0.000560	0.022506	0.005968
Avg_Credit_Limit	0.677962	0.068604	1.000000	0.608860	-0.100312	0.551385	-0.414352
Total_Credit_Cards	0.739329	-0.010281	0.608860	1.000000	0.315796	0.167758	-0.651251
Total_visits_bank	0.406438	-0.000560	-0.100312	0.315796	1.000000	-0.551861	-0.506016
Total_visits_online	0.033916	0.022506	0.551385	0.167758	-0.551861	1.000000	0.127299
Total_calls_made	-0.684125	0.005968	-0.414352	-0.651251	-0.506016	0.127299	1.000000

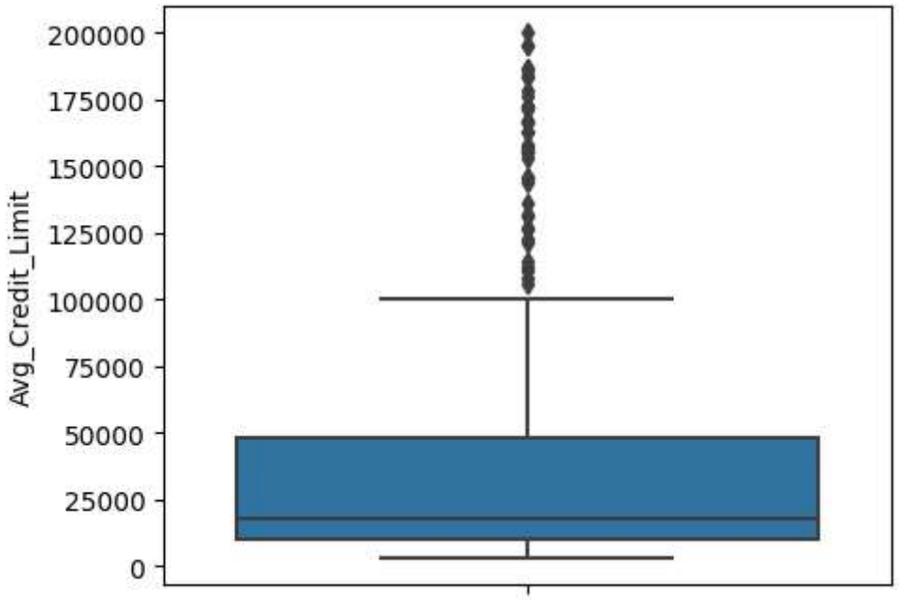
In [8]:
```
plt.figure(figsize=(10,8))
sns.heatmap(corr_matrix)
plt.show()
```

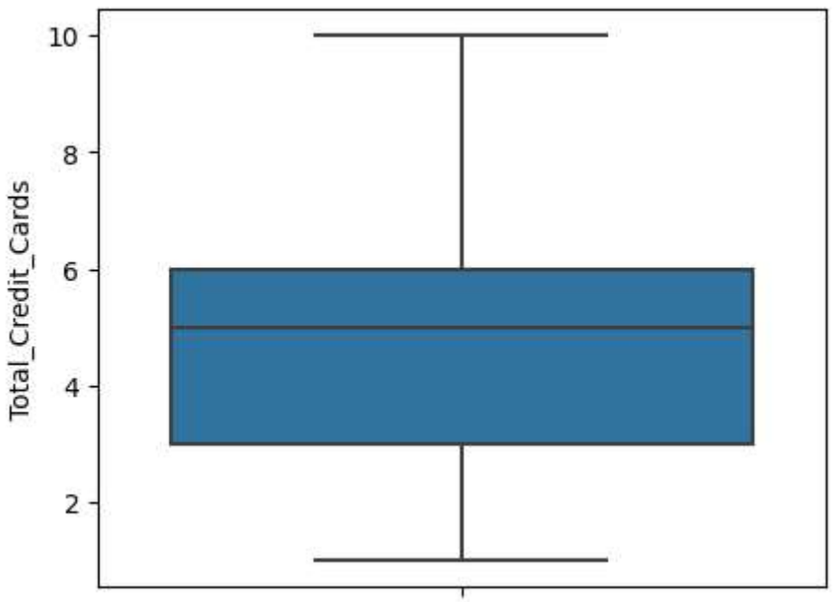


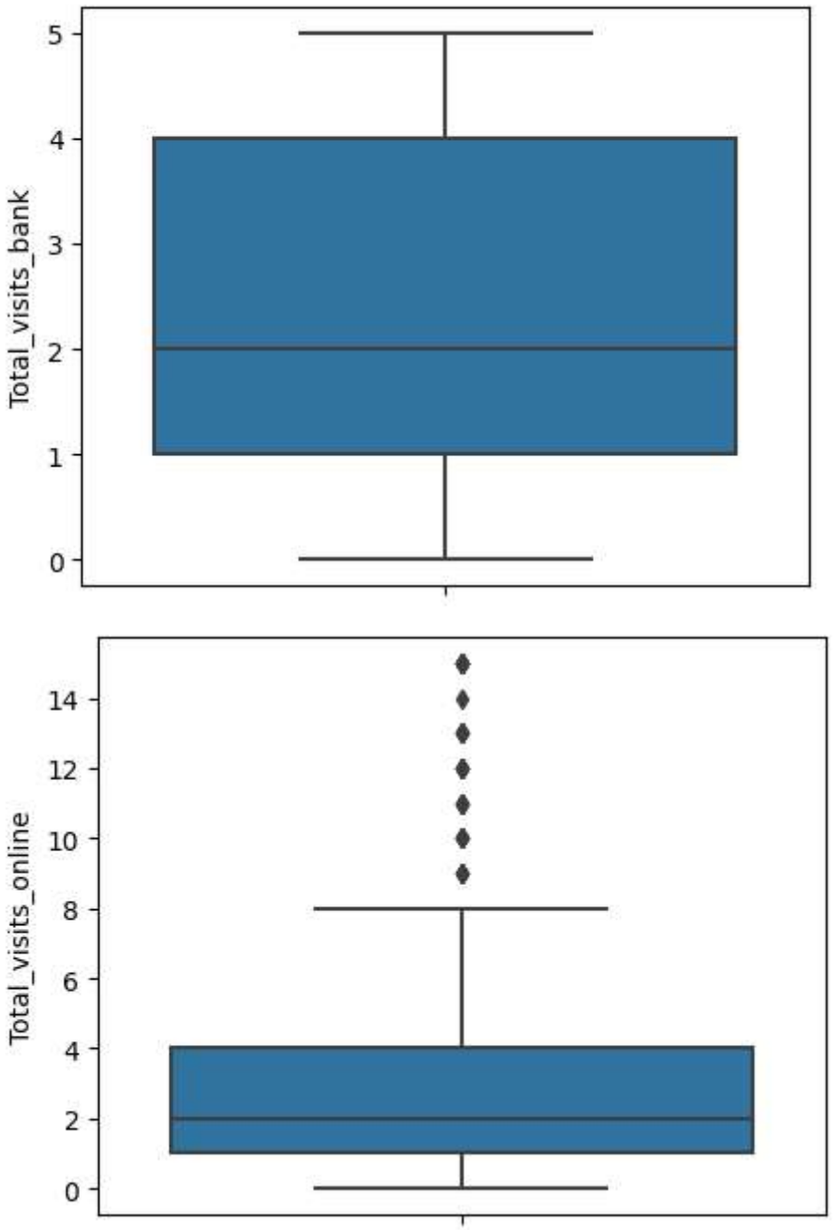
In [9]:
```
for cols in df.select_dtypes(include=np.number).columns:
    plt.figure(figsize=(5,4))
    sns.boxplot(y=df[cols])
    plt.show()
```

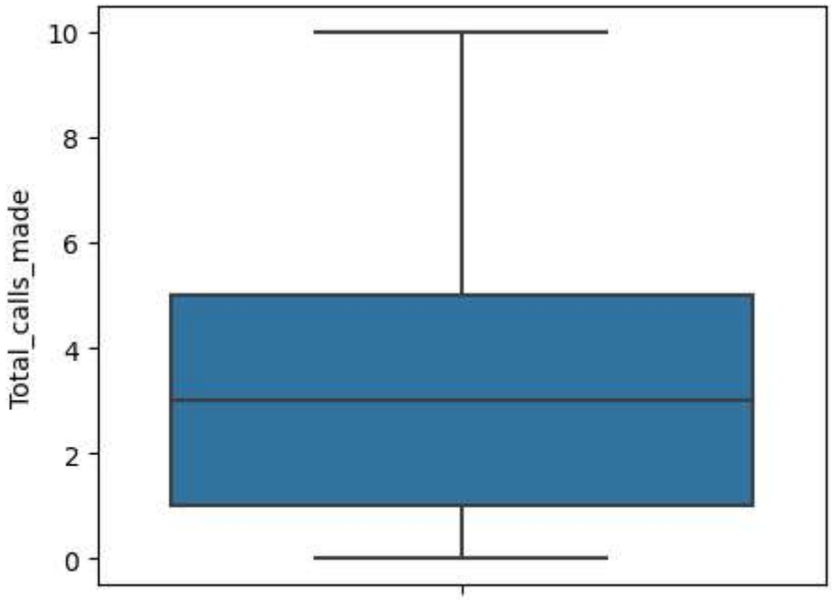





Avg_Credit_Limit
200000
175000
150000
125000
100000
75000
50000
25000
0


Total_Credit_Cards
10
8
6
4
2

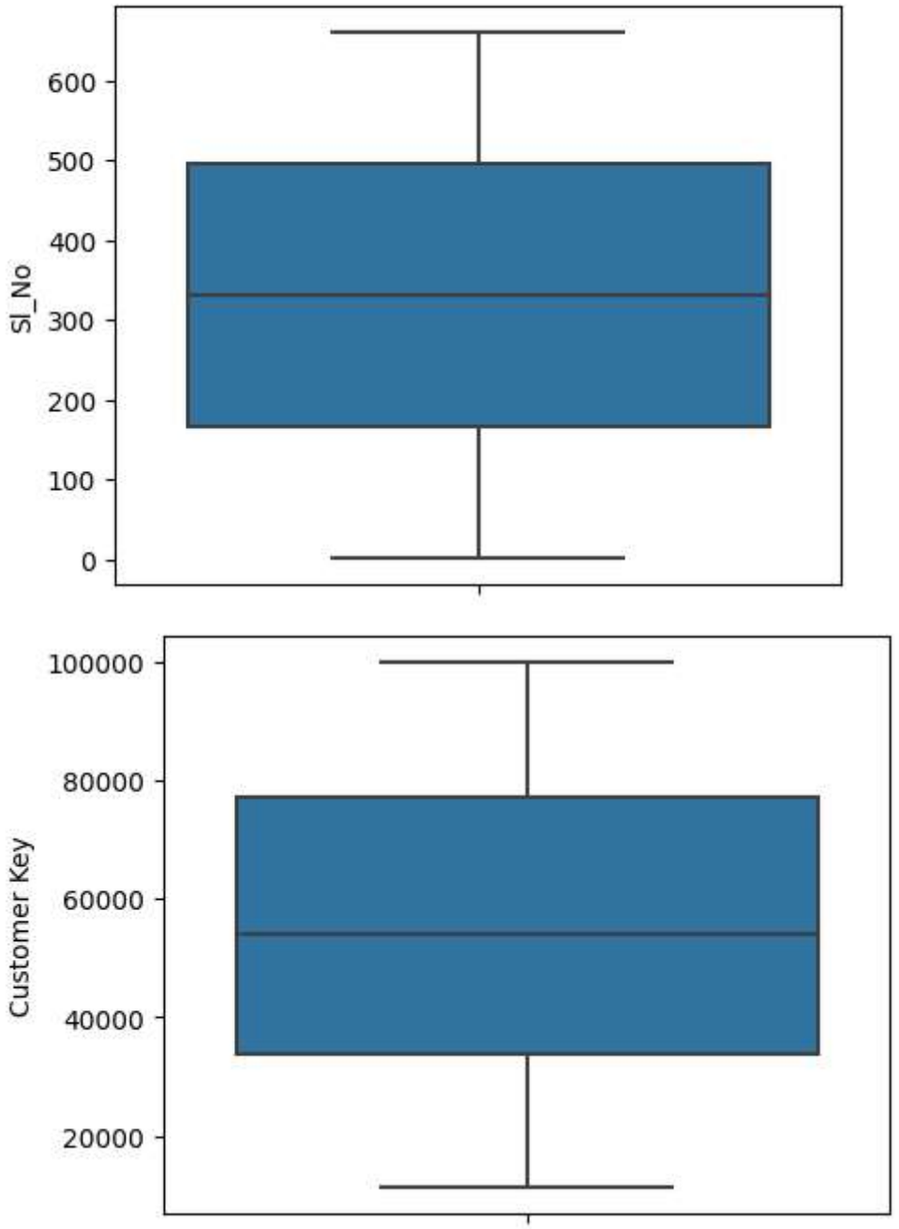

Total_visits_bank
0
1
2
3
4
5
Total_visits_online
0
2
4
6
8
10
12
14

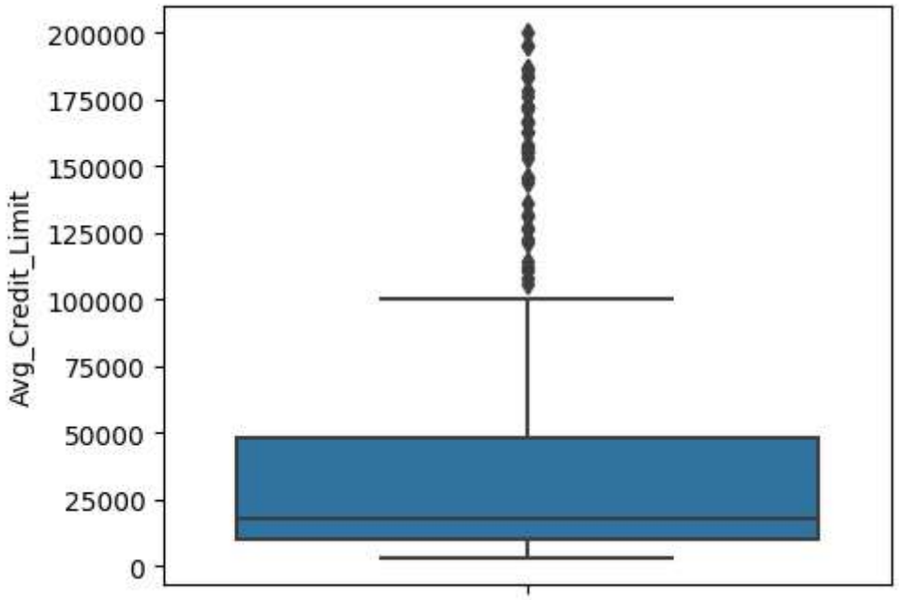


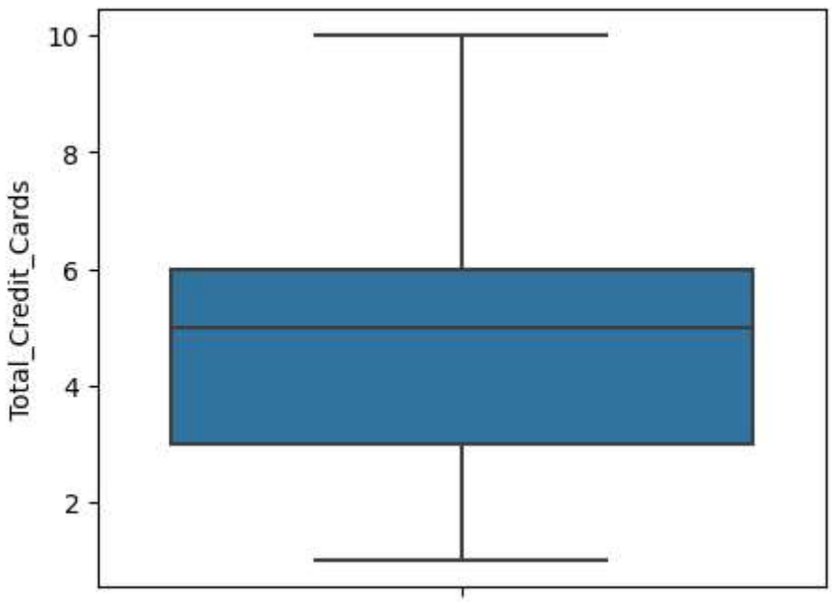
In [10]:
```
def drop_outliers(df, field):
    q3 = df[field].quantile(0.75)
    q1 = df[field].quantile(0.25)
    iqr = q3 - q1
    df.drop(df[df[field] > (q3 + 1.5 * iqr)].index, inplace = True)
    df.drop(df[df[field] < (q1 - 1.5 * iqr)].index, inplace = True)
```

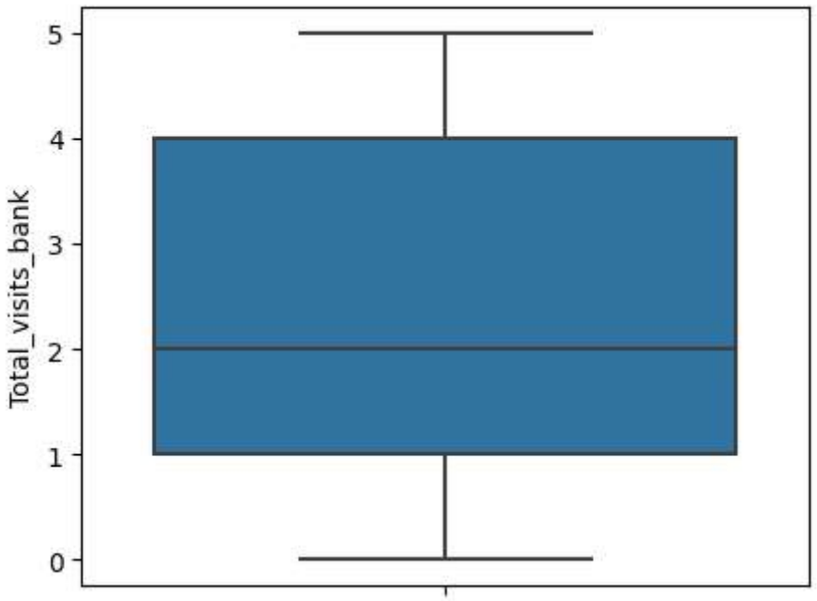
In [11]:
```
outliers_cols = ['Avg_Credit_Limit', 'Total_visits_online']

for col in outliers_cols:
    drop_outliers(df, cols)
```

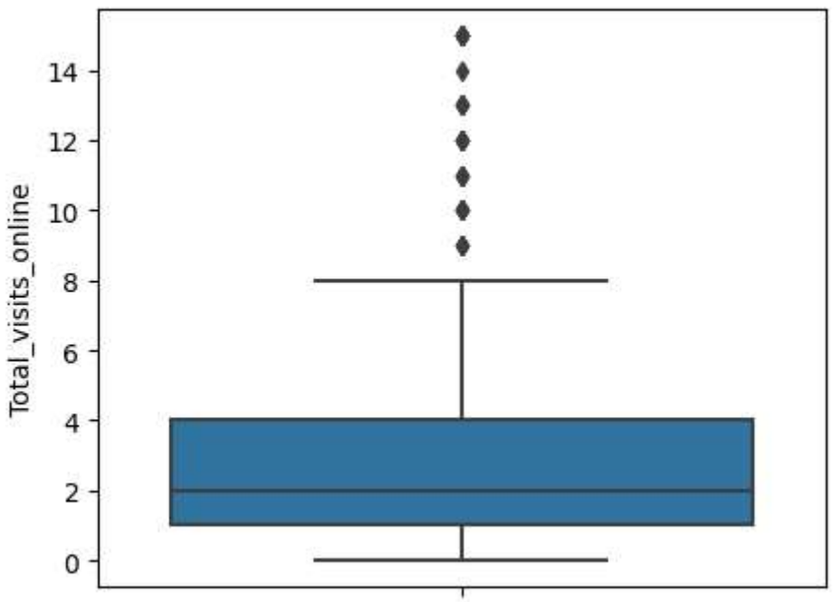
In [12]:
```
for cols in df.select_dtypes(include=np.number).columns:
    plt.figure(figsize=(5,4))
    sns.boxplot(y=df[cols])
    plt.show()
```

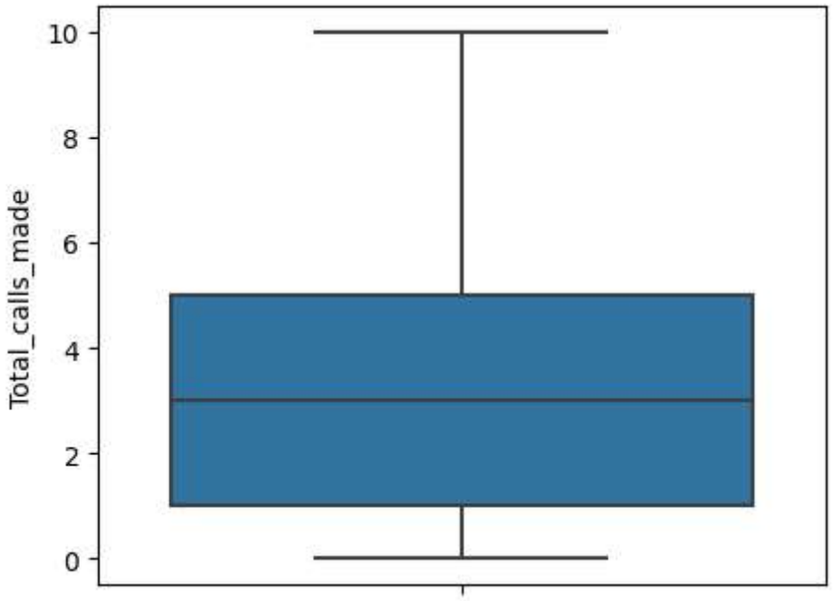

Sl_No
600
500
400
300
200
100
0
Customer Key
100000
80000
60000
40000
20000


Avg_Credit_Limit
200000
175000
150000
125000
100000
75000
50000
25000
0


Total_Credit_Cards
10
8
6
4
2


Total_visits_bank
5
4
3
2
1
0

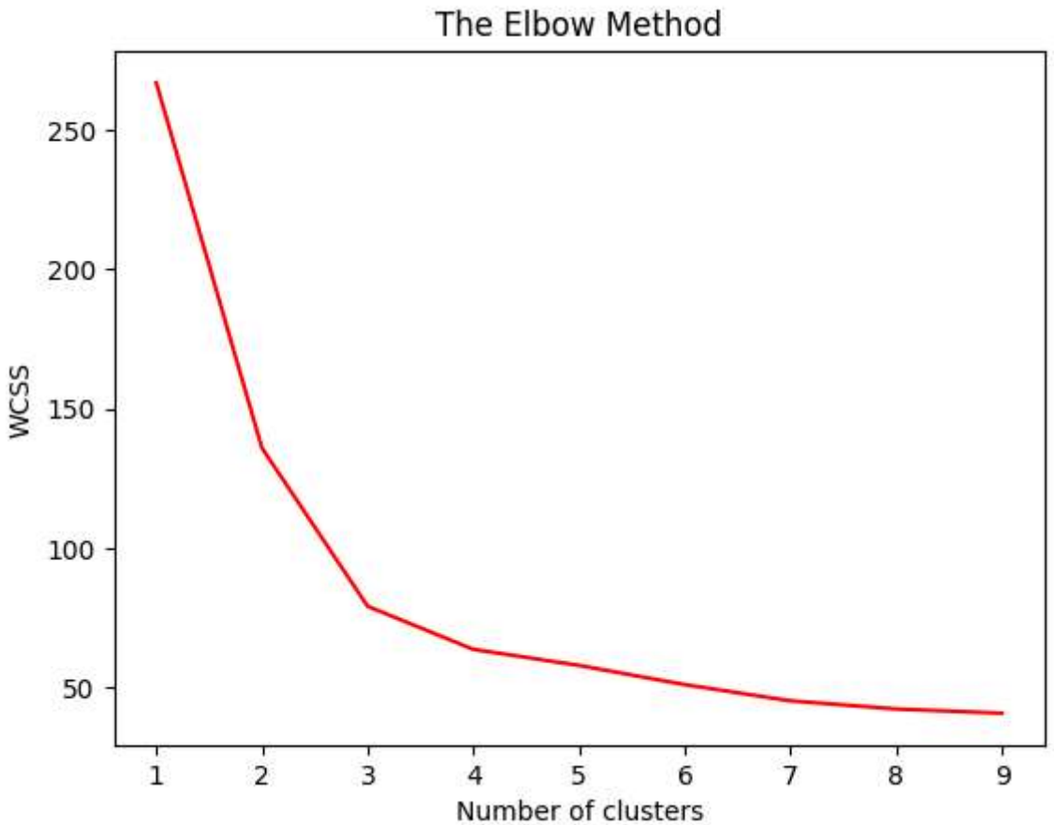

Total_visits_online
14
12
10
8
6
4
2
0



K-means Clustering

In [13]:
```
import warnings
warnings.filterwarnings('ignore')

X = df.drop(['Sl_No', 'Customer Key'], axis=1).reset_index()

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X = scaler.fit_transform(X)

wcss = []
for i in range(1,10):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.plot(range(1,10), wcss, color = 'red')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
File "C:\Users\Sushant\AppData\Local\Programs\Python\Python310\lib\site-packages\joblib\externals\loky\backend\context.py", line 282, in _count_physical_cores
  raise ValueError(f"found {cpu_count_physical} physical cores < 1")
```



In [14]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters = 3)
kmeans = kmeans.fit(X)
labels = kmeans.predict(X)
centroids = kmeans.cluster_centers_
df['Cluster'] = labels
df.head()
```

Out[14]:

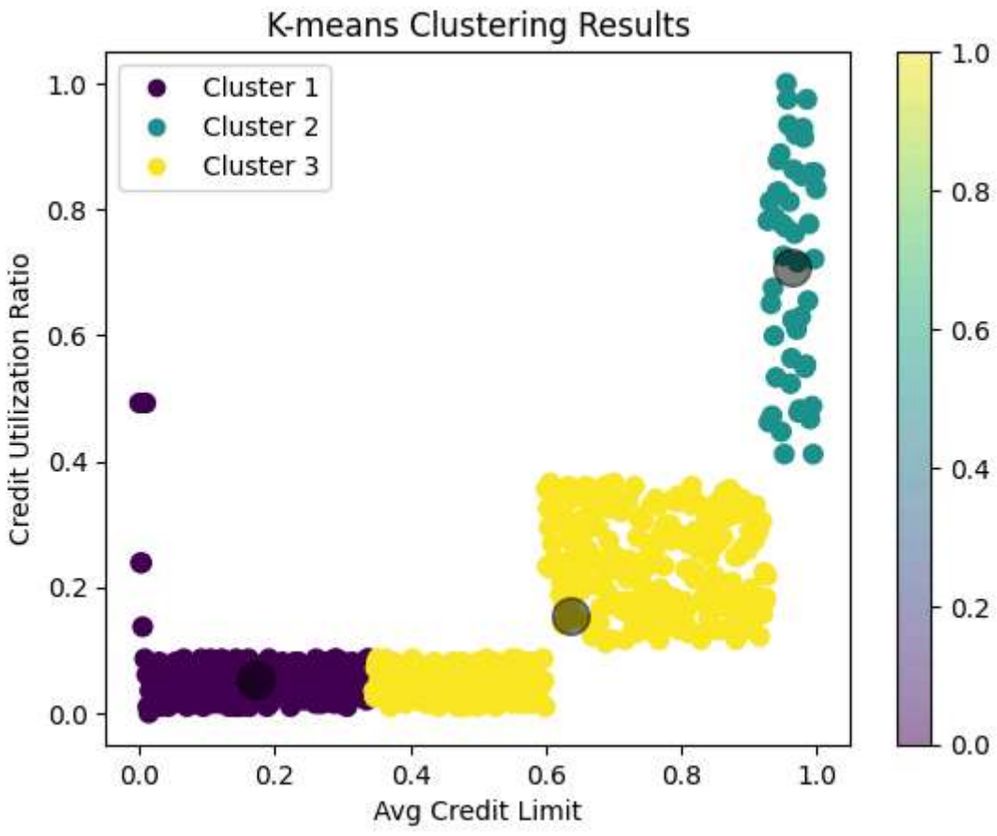
	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made	Cluster
0	1	87073	100000	2	1	1	0	0
1	2	38414	50000	3	0	10	9	0
2	3	17341	50000	7	1	3	4	0
3	4	40496	30000	5	1	1	4	0
4	5	47437	100000	6	0	12	3	0

In [15]:
```
centroids
```

Out[15]:
```
array([[0.17223065, 0.0537893 , 0.1622807 , 0.18421053, 0.24035088,
        0.67894737],
       [0.96433991, 0.70939086, 0.875     , 0.125     , 0.725     ,
        0.10208333],
       [0.63657056, 0.15486199, 0.50202546, 0.70104167, 0.06527778,
        0.2       ]])
```

In [16]:
```
scatter = plt.scatter(X[:,0], X[:,1],c=labels, s=50, cmap ='viridis')

plt.scatter(centroids[:,0], centroids[:,1],c='black', s=200, alpha = 0.5)
plt.legend(handles=scatter.legend_elements()[0], labels = ['Cluster 1','Cluster 2','Cluster 3'])
plt.xlabel('Avg Credit Limit')
plt.ylabel('Credit Utilization Ratio')
plt.title('K-means Clustering Results')
plt.colorbar()
```

Out[16]: <matplotlib.colorbar.Colorbar at 0x1a1b6c83df0>



In [ ]: