Credit Card Clustering

Library Importing

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Data Import

In [53]:
```
data=pd.read_csv(r'C:\Users\DELL\Desktop\Data\CC GENERAL.csv')
```

In [54]:
```
data
```

Out[54]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PUR
0	C10001	40.900749	0.818182	95.40	0.00	
1	C10002	3202.467416	0.909091	0.00	0.00	
2	C10003	2495.148862	1.000000	773.17	773.17	
3	C10004	1666.670542	0.636364	1499.00	1499.00	
4	C10005	817.714335	1.000000	16.00	16.00	
...	...	...	...	...	...	
8945	C19186	28.493517	1.000000	291.12	0.00	
8946	C19187	19.183215	1.000000	300.00	0.00	
8947	C19188	23.398673	0.833333	144.40	0.00	
8948	C19189	13.457564	0.833333	0.00	0.00	
8949	C19190	372.708075	0.666667	1093.25	1093.25	

8950 rows × 18 columns

Data Reading

In [55]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   CUST_ID                           8950 non-null   object 
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64  
 12  PURCHASES_TRX                     8950 non-null   int64  
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64  
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

In [56]:
```
data.shape
```

Out[56]:
```
(8950, 18)
```

Treat Missing Values

In [57]:
```
data=data.interpolate(method='pad')
```

In [58]:
```
data.isnull().sum()
```

Out[58]:
```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        0
PAYMENTS                            0
MINIMUM_PAYMENTS                    0
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

Variable Selection

**There are three features in the dataset which are very important for the task of credit card segmentation:

1.BALANCE: The balance left in the accounts of credit card customers.

2.PURCHASES: Amount of purchases made from the accounts of credit card customers.

3.CREDIT_LIMIT: The limit of the credit card.

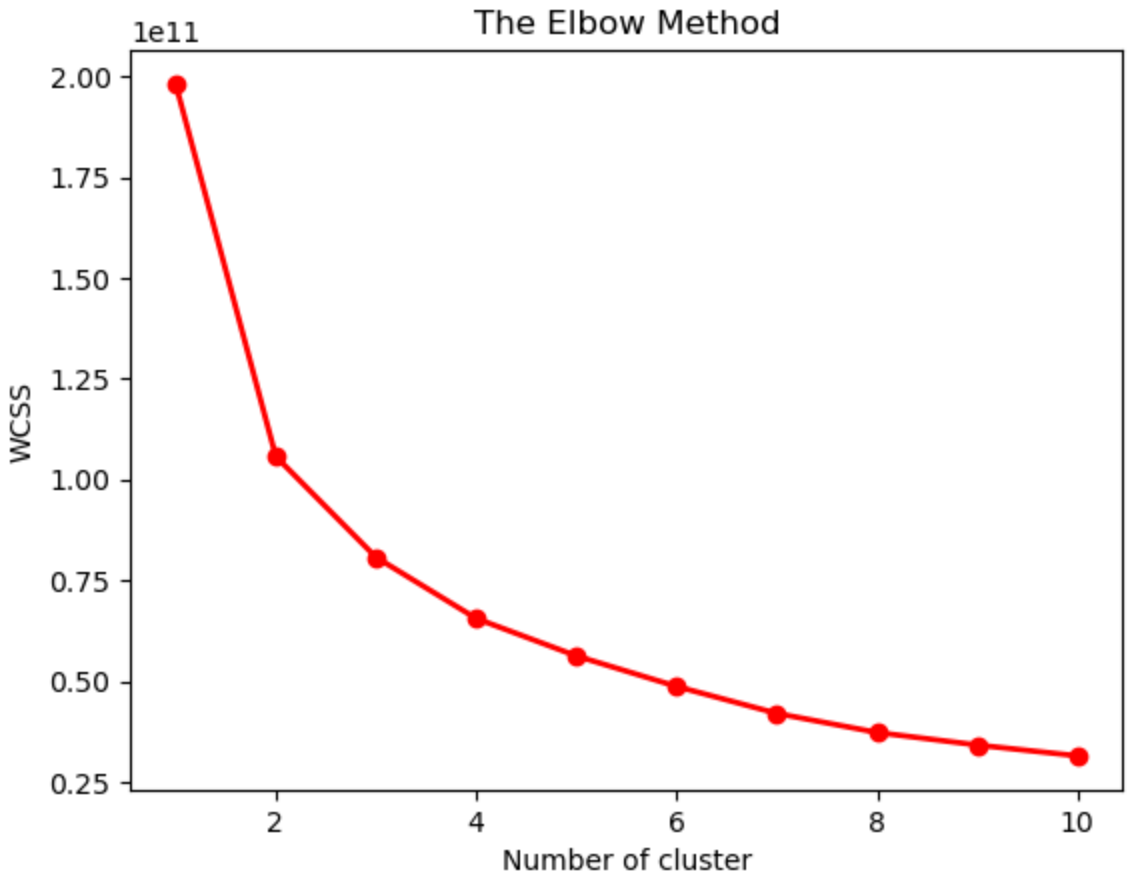
These three features are enough to group credit card holders as they tell us about the buying history, bank balance, and credit limit of the credit card holders. So let's use these 3 features to create clusters from the dataset**

In [59]:
```
X= data.iloc[:,[1,3,-5]].values
print(X)
```

```
[[  40.900749   95.4      1000.      ]
 [3202.467416    0.       7000.      ]
 [2495.148862  773.17     7500.      ]
 ...
 [  23.398673  144.4      1000.      ]
 [  13.457564    0.        500.      ]
 [ 372.708075 1093.25     1200.      ]]
```

Creating K-Mean Clustering Model

In [60]:
```
import warnings
warnings.filterwarnings("ignore")
from sklearn.cluster import KMeans
wcss=[]
for i in range (1,11):
    Kmeans =KMeans(n_clusters=i,init='k-means++',random_state=0)  #we use K-means++ for
    Kmeans.fit(X)
    wcss.append(Kmeans.inertia_) #inertia_ is taken for formula(WCSS)
plt.plot(range(1,11),wcss, linewidth=2, markersize=12,marker='.',color = 'red')
plt.title('The Elbow Method')
plt.xlabel('Number of cluster')
plt.ylabel('WCSS')
plt.show()
```



In [61]:
```
Kmeans =KMeans(n_clusters=5,init='k-means++',random_state=0)
y_kmeans= Kmeans.fit_predict(X)
print(y_kmeans)
```

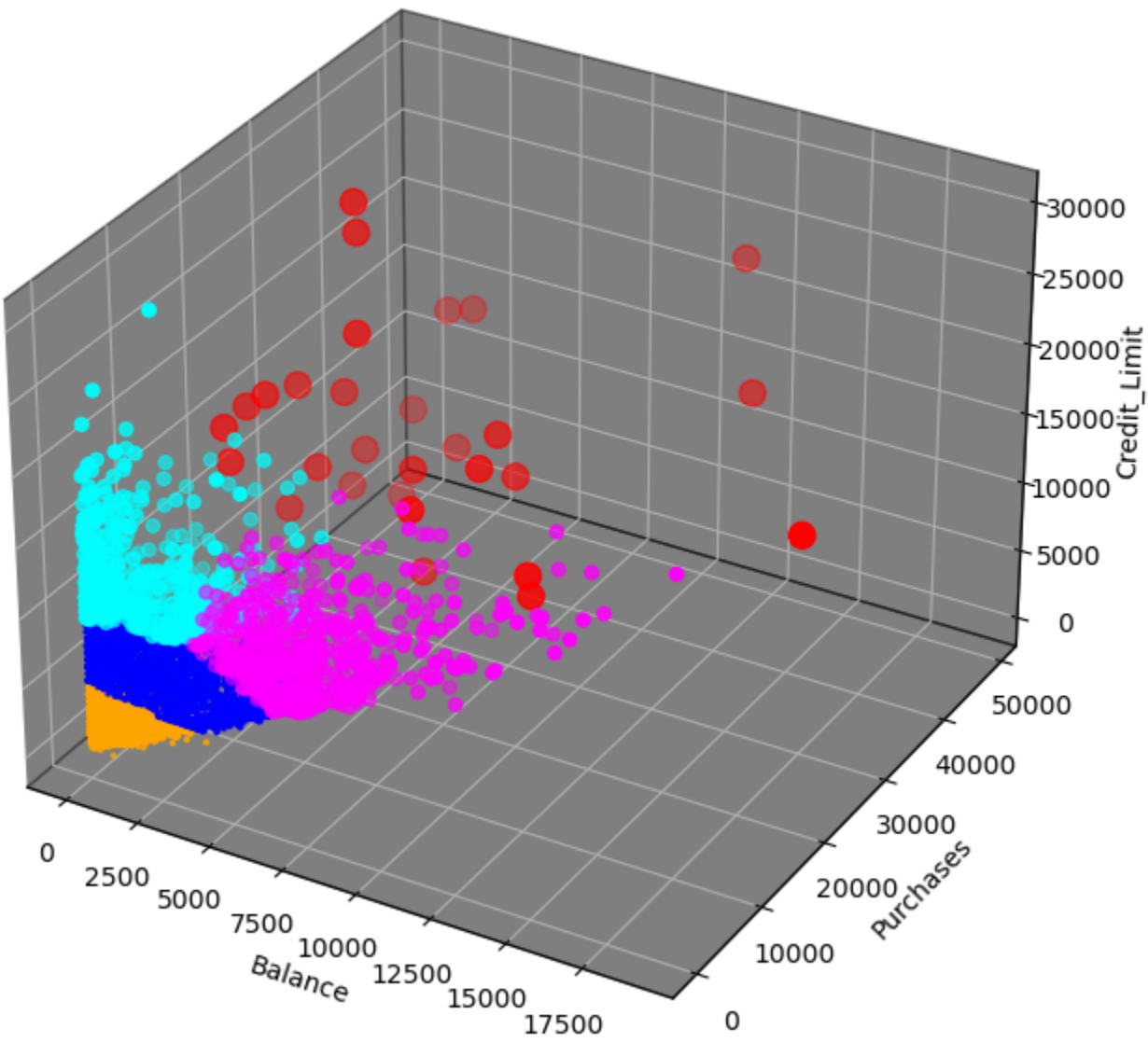
```
[2 0 0 ... 2 2 2]
```

Visualization

In [62]:
```
fig = plt.figure(figsize = (8,8))
ax= fig.add_subplot(111, projection='3d')
ax.xaxis.pane.set_color('black')
ax.yaxis.pane.set_color('black')
ax.zaxis.pane.set_color('Black')
ax.xaxis.pane.set_edgecolor('black')
ax.yaxis.pane.set_edgecolor('black')
ax.zaxis.pane.set_edgecolor('black')

ax.scatter(X[y_kmeans==0,0],X[y_kmeans==0,1],X[y_kmeans==0,2], s=10,c='blue', marker= '.
ax.scatter(X[y_kmeans==1,0],X[y_kmeans==1,1],X[y_kmeans==1,2],s=100,c='magenta', marker=
ax.scatter(X[y_kmeans==2,0],X[y_kmeans==2,1],X[y_kmeans==2,2],s=10,c='orange', marker= '
ax.scatter(X[y_kmeans==3,0],X[y_kmeans==3,1],X[y_kmeans==3,2],s=100,c='cyan', marker= '.
ax.scatter(X[y_kmeans==4,0],X[y_kmeans==4,1],X[y_kmeans==4,2],s=100,c='red', marker= 'o'

ax.set_title('Credit Card Clustering')
ax.set_xlabel('Balance')
ax.set_ylabel('Purchases')
ax.set_zlabel('Credit_Limit')
plt.show()
```



In [ ]: