```
from google.colab import files
uploaded = files.upload()
```

Choose files CC GENERAL.csv

- **CC GENERAL.csv**(text/csv) - 902879 bytes, last modified: 08/07/2025 - 100% done

```
Saving CC GENERAL.csv to CC GENERAL.csv
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

```
df = pd.read_csv("CC GENERAL.csv")
print("Shape:", df.shape)
df.head()
```

```
Shape: (8950, 18)
```

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PU
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	

Next steps: Generate code with df View recommended plots New interactive sheet

BALANCE BALANCE_FREQUENCY PURCHASES

ONEOFF_PURCHASES

BALANCE vs BALANCE_FREQUENCY

BALANCE_FREQUENCY vs PURCHASES

PURCHASES vs ONEOFF_PURCHASES

ONEOFF_PURCHASES vs INSTALLMENTS_PURCHASES

BALANCE

BALANCE_FREQUENCY

PURCHASES

ONEOFF_PURCHASES

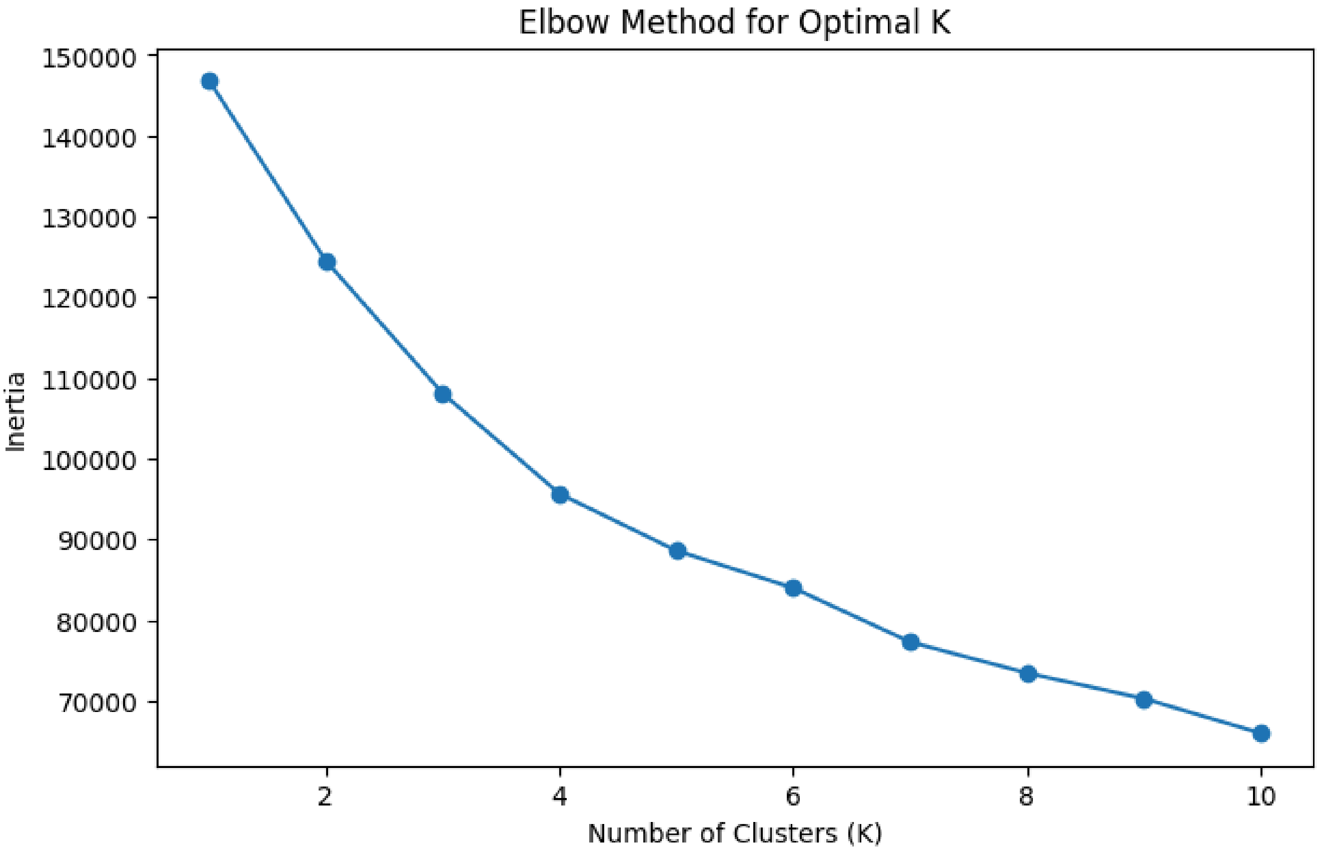
```
# Drop CUST_ID column (not useful for clustering)
df.drop('CUST_ID', axis=1, inplace=True)

# Handle missing values
df.dropna(inplace=True)

print("After cleaning:", df.shape)
```

```
After cleaning: (8636, 17)
```

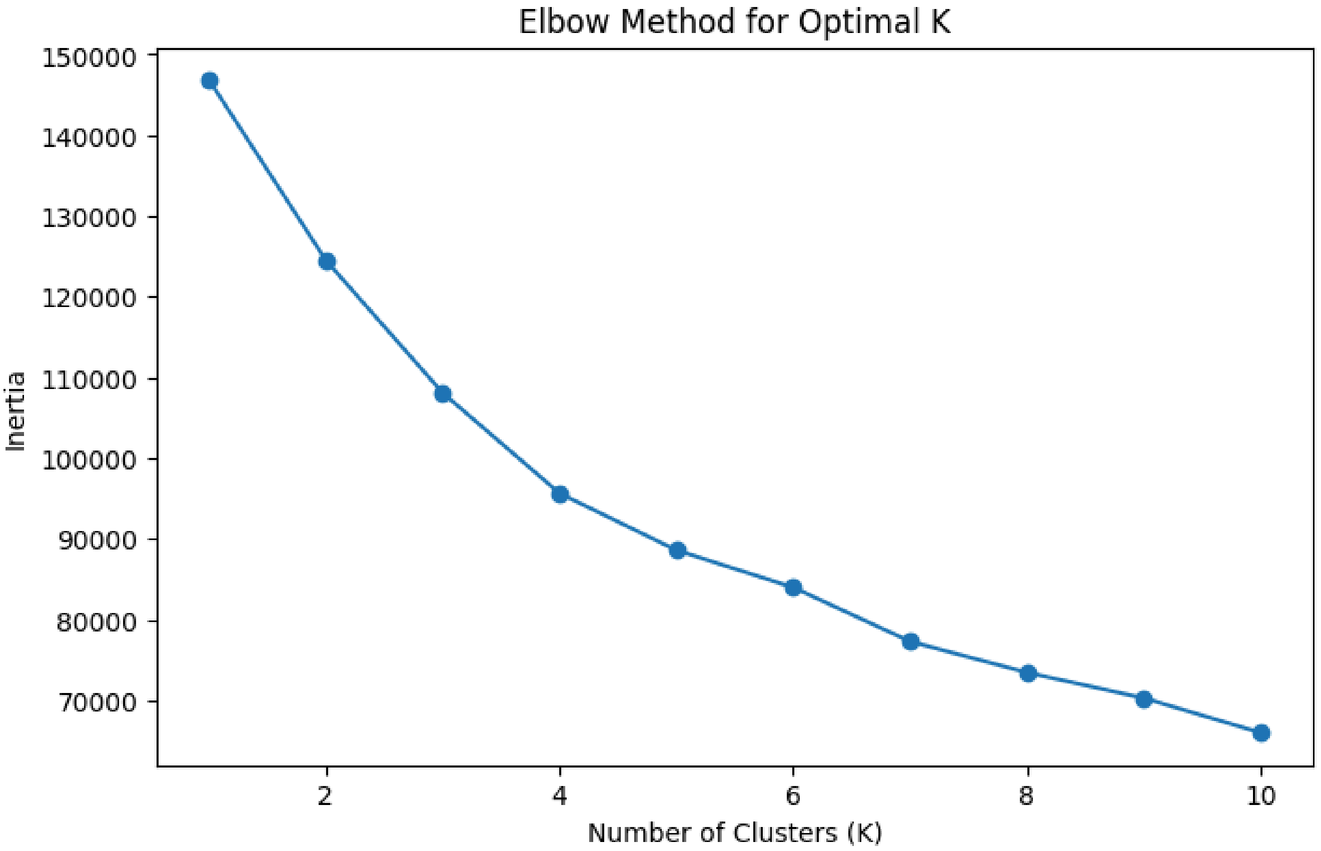
```
scaler = StandardScaler()
scaled_df = scaler.fit_transform(df)
```

```
inertia = []
K = range(1, 11)
```

```
for k in K:
    model = KMeans(n_clusters=k, random_state=42)
    model.fit(scaled_df)
    inertia.append(model.inertia_)

plt.figure(figsize=(8,5))
plt.plot(K, inertia, marker='o')
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Inertia")
plt.title("Elbow Method for Optimal K")
plt.show()
```

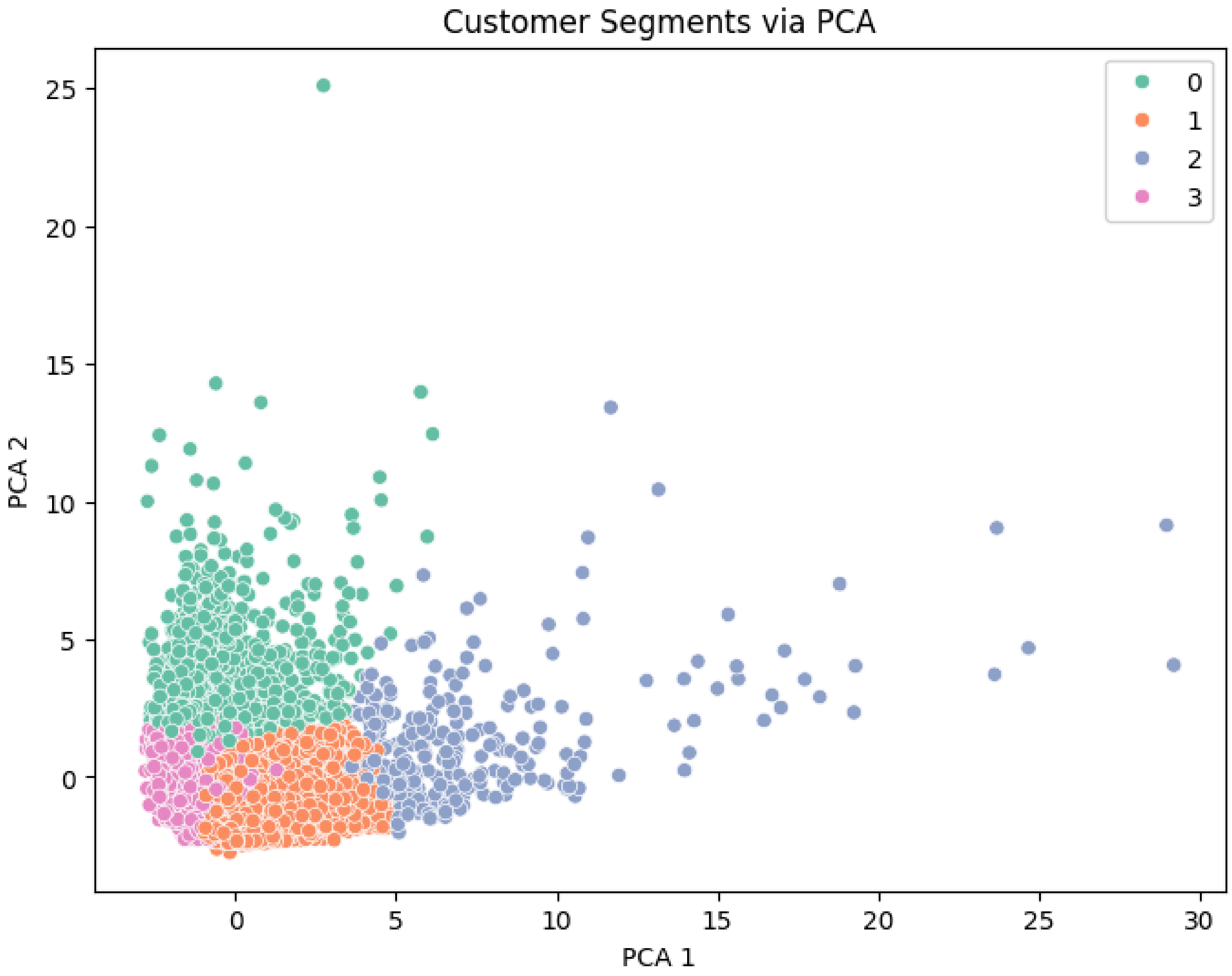


```
inertia = []
K = range(1, 11)

for k in K:
    model = KMeans(n_clusters=k, random_state=42)
    model.fit(scaled_df)
    inertia.append(model.inertia_)

plt.figure(figsize=(8,5))
plt.plot(K, inertia, marker='o')
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Inertia")
plt.title("Elbow Method for Optimal K")
plt.show()
```



```
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(scaled_df)

# Add cluster labels to original data
df['Cluster'] = clusters
```

```
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(scaled_df)

plt.figure(figsize=(8,6))
sns.scatterplot(x=reduced_data[:,0], y=reduced_data[:,1], hue=clusters, palette="Set2")
plt.title("Customer Segments via PCA")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.show()
```



```
df.groupby('Cluster').mean()
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PUR
Cluster					
0	4652.097509	0.969074	511.046007	324.079611	187.
1	970.417580	0.950767	1374.131996	687.243458	687.
2	3941.953414	0.985355	8980.111024	5968.520137	3013.
3	1052.425406	0.818274	277.900840	212.097638	66.

Start coding or generate with AI.