PRACTICAL NO. 8

AIM: K-Means Clustering

- Apply the K-Means algorithm to group similar data points into clusters.
- Determine the optimal number of clusters using elbow method or silhouette analysis.
- Visualize the clustering results and analyze the cluster characteristics.

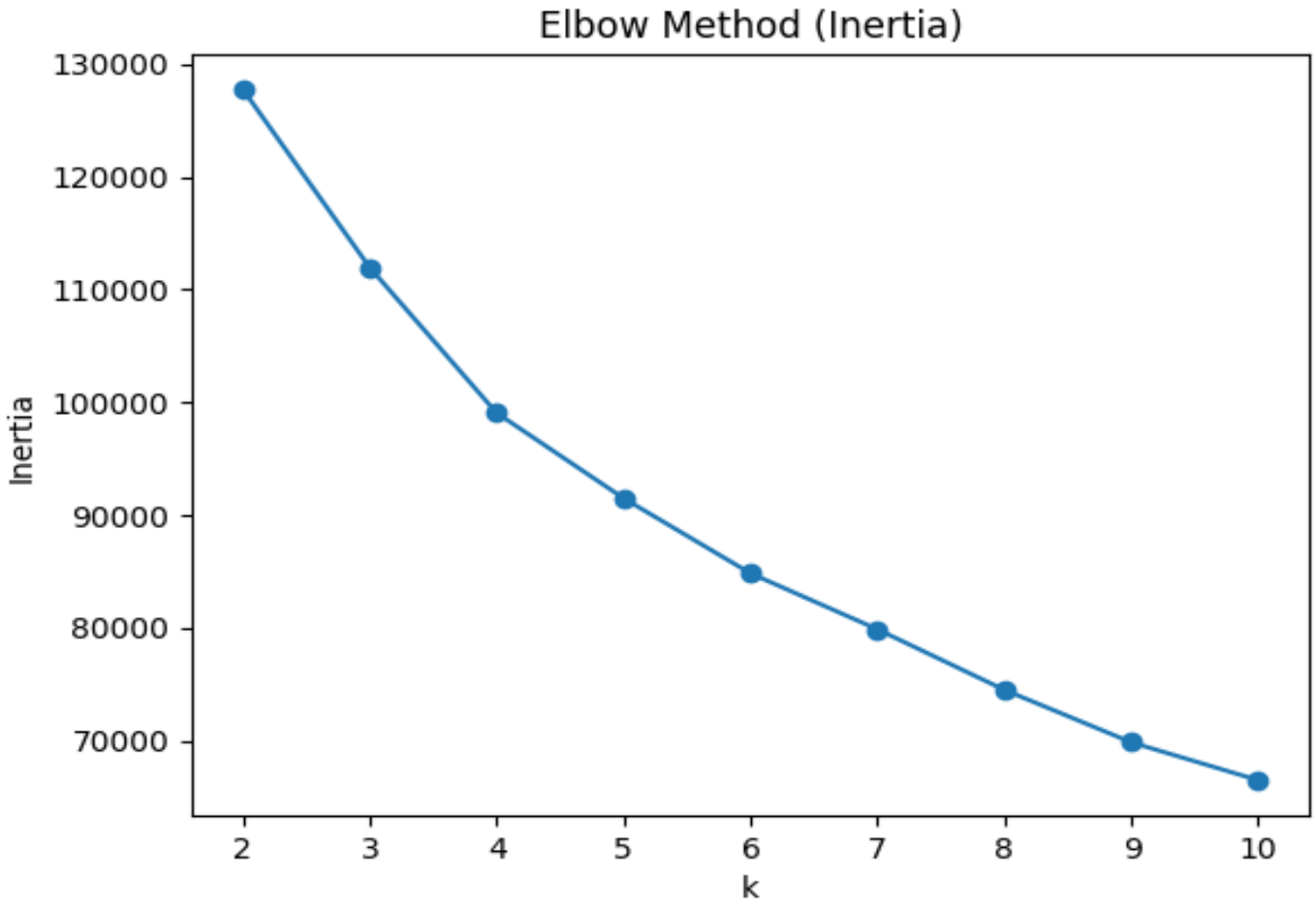
CODE:-

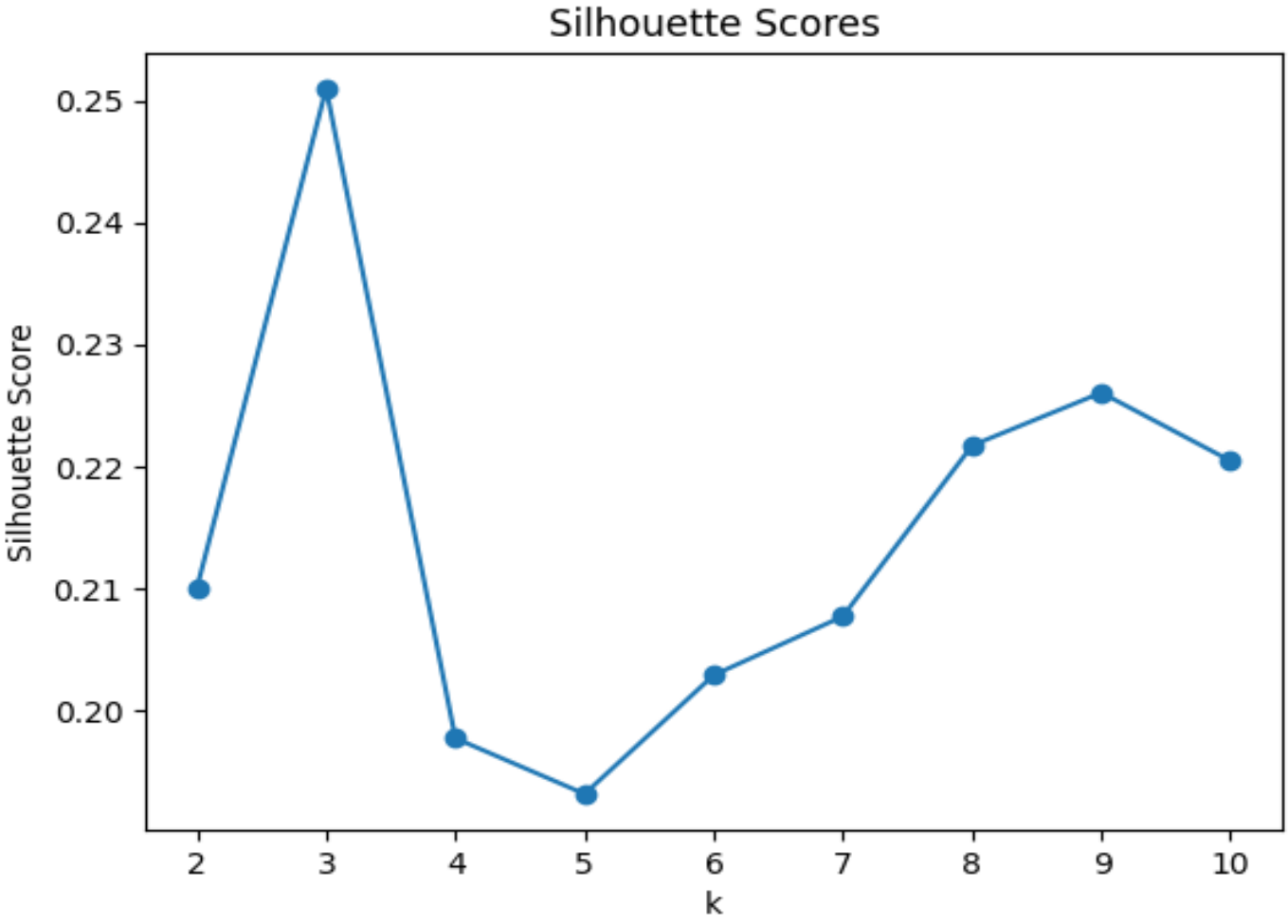
```
%matplotlib inline
import pandas as pd, numpy as np, matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans, MiniBatchKMeans
from sklearn.metrics import silhouette_score
pd.set_option('display.max_columns', 200)
pd.set_option('display.width', 200)

DATA_PATH = "CC GENERAL.csv"
OUT_CLUSTERED = "cc_general_with_clusters.csv"
OUT_PROFILE = "cluster_profile.csv"

# ---- Load + quick EDA ----
df = pd.read_csv(DATA_PATH)
print("Loaded:", DATA_PATH, "shape:", df.shape)
print("\nColumns:\n", df.columns.tolist())
print("\nMissing values per column:\n", df.isna().sum())

# ---- Preprocess ----
df_model = df.copy()
# drop obvious id columns
id_candidates = [c for c in df_model.columns if c.lower() in
('cust_id','customerid','clientnum','customer_id','client_id','id')]
if id_candidates:
    print("Dropping id-like columns:", id_candidates)
    df_model = df_model.drop(columns=id_candidates)

# keep numeric only
non_numeric = df_model.select_dtypes(exclude=[np.number]).columns.tolist()
if non_numeric:
    print("Dropping non-numeric cols:", non_numeric)
df_model = df_model.select_dtypes(include=[np.number])

# impute & scale
imputer = SimpleImputer(strategy='median')
X_imputed = imputer.fit_transform(df_model)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)
print("\nPreprocessing done. Numeric features used:", df_model.shape[1])

# ---- Choose k via Elbow + Silhouette (k=2..10) ----
inertias, sil_scores = [], []
K_range = range(2, 11)
```

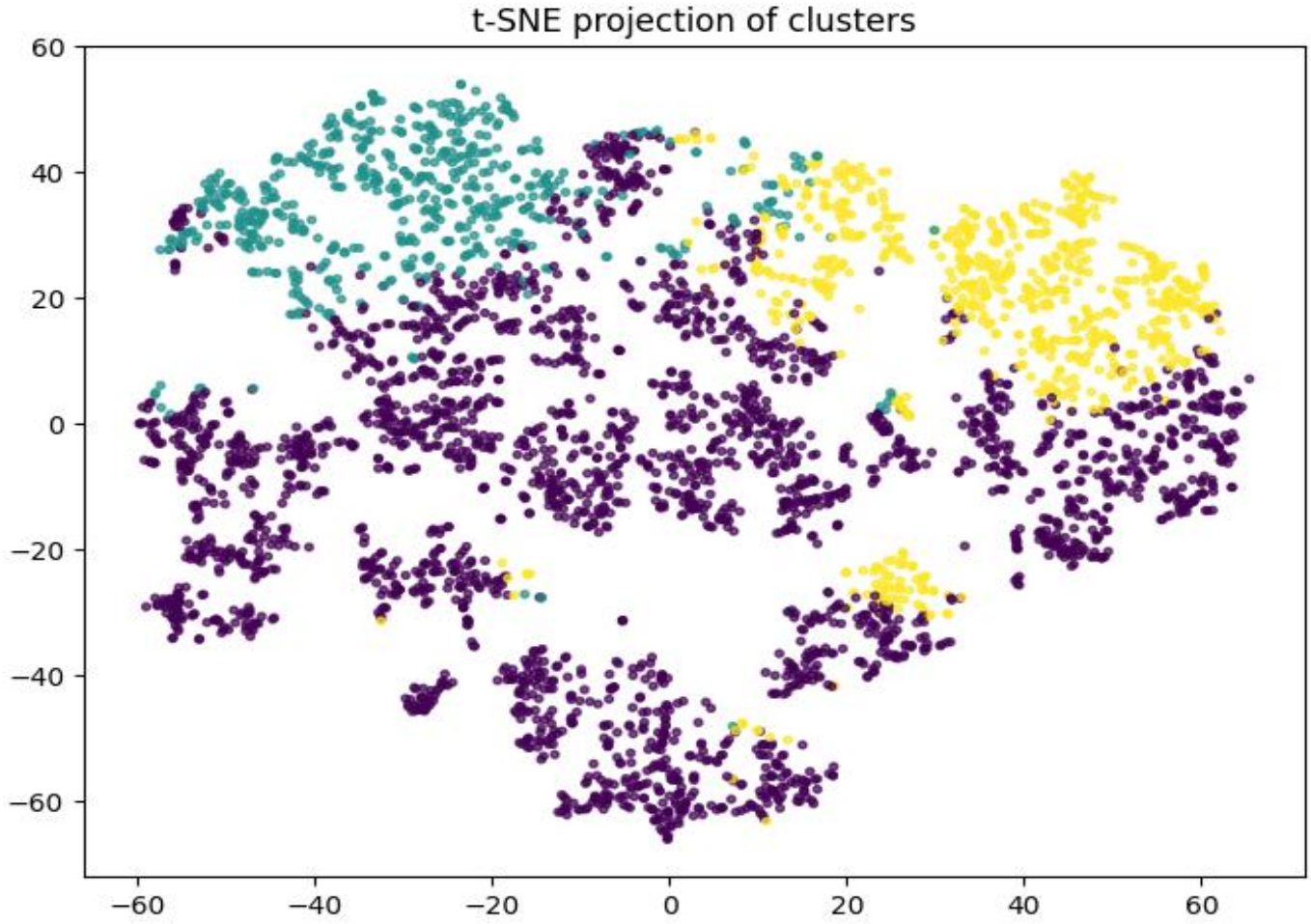
```
for k in K_range:
    km = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = km.fit_predict(X_scaled)
    inertias.append(km.inertia_)
    sil_scores.append(silhouette_score(X_scaled, labels))

best_k = K_range[int(np.argmax(sil_scores))]

# ---- ADDED: Elbow & Silhouette plots (no figsize, default sizing preserved) ----
# Elbow plot (Inertia vs k)
plt.plot(list(K_range), inertias, marker='o')
plt.title("Elbow Method (Inertia)")
plt.xlabel("k")
plt.ylabel("Inertia")
plt.show()

# Silhouette plot (Silhouette Score vs k)
plt.plot(list(K_range), sil_scores, marker='o')
plt.title("Silhouette Scores")
plt.xlabel("k")
plt.ylabel("Silhouette Score")
plt.show()
# ---- end added plots ----

print("\nInertias:", dict(zip(K_range, [round(i,2) for i in inertias])))
print("Silhouette scores:", dict(zip(K_range, [round(s,3) for s in sil_scores])))
print("Chosen k (max silhouette):", best_k)

# ---- Fit final clusters ----
kmeans = KMeans(n_clusters=best_k, random_state=42, n_init=20).fit(X_scaled)
labels_km = kmeans.labels_
mbk = MiniBatchKMeans(n_clusters=best_k, random_state=42, batch_size=256).fit(X_scaled)
labels_mbk = mbk.labels_

# attach to df and save
df_results = df.copy()
df_results['cluster_kmeans'] = labels_km
df_results['cluster_minibatch'] = labels_mbk
df_results.to_csv(OUT_CLUSTERED, index=False)
print(f"\nSaved clustered data -> {OUT_CLUSTERED}")

# ---- Print cluster counts ----
print("\nCluster counts (KMeans):")
print(df_results['cluster_kmeans'].value_counts().sort_index().to_string())
print("\nCluster counts (MiniBatchKMeans):")
print(df_results['cluster_minibatch'].value_counts().sort_index().to_string())

# ---- PCA 2D + centroids ----
pca2 = PCA(n_components=2, random_state=42)
proj2 = pca2.fit_transform(X_scaled)
centroids_pca = pca2.transform(kmeans.cluster_centers_)

plt.figure(figsize=(8,6))
plt.scatter(proj2[:,0], proj2[:,1], c=labels_km, s=12, cmap='viridis', alpha=0.7)
plt.scatter(centroids_pca[:,0], centroids_pca[:,1], marker='X', s=200, c='red', edgecolor='black', linewidth=1.5)
```

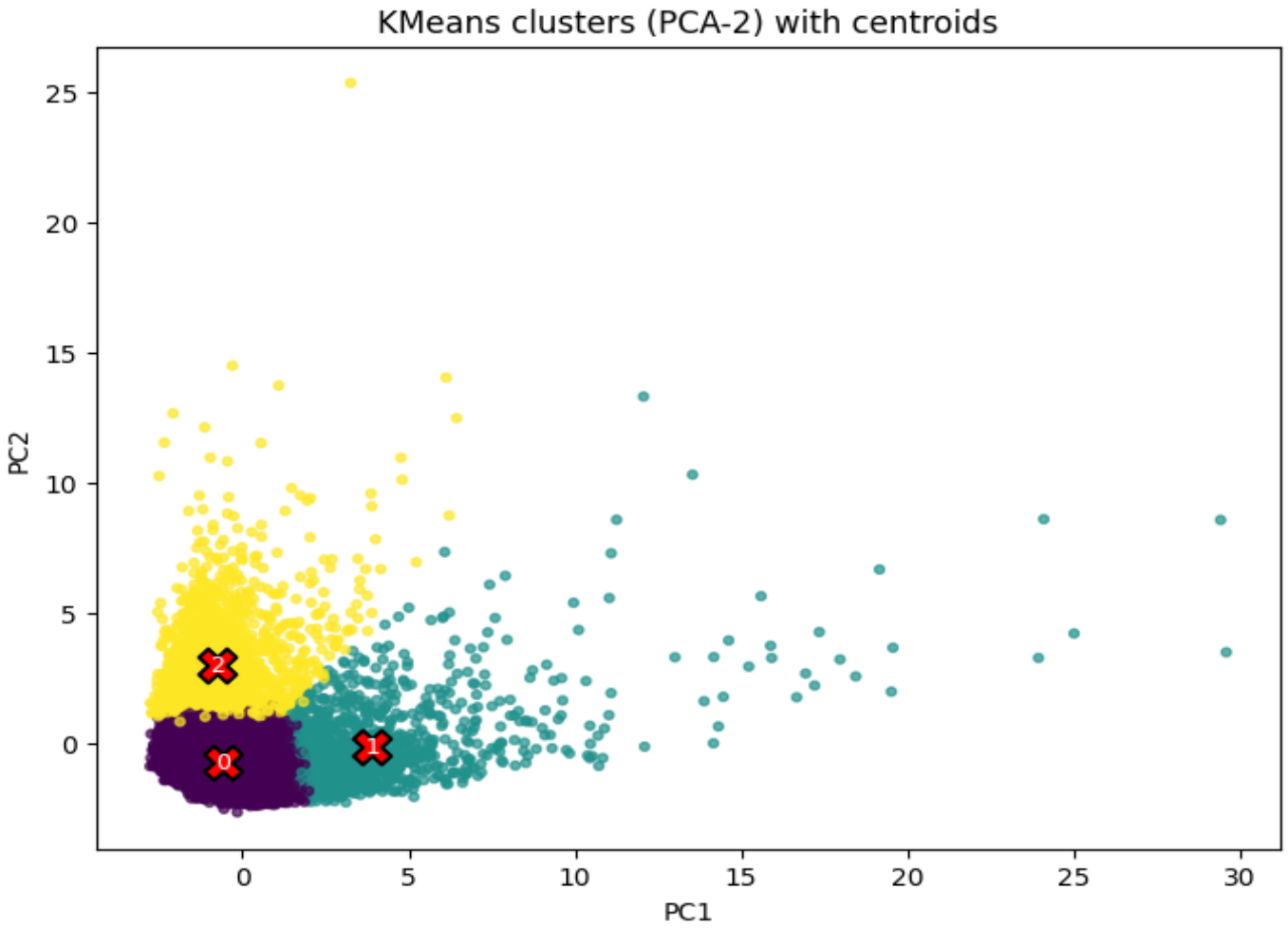
```
for i,(x,y) in enumerate(centroids_pca):
    plt.text(x, y, str(i), color='white', fontsize=9, ha='center', va='center')
plt.title("KMeans clusters (PCA-2) with centroids")
plt.xlabel("PC1"); plt.ylabel("PC2"); plt.show()

# ---- t-SNE (subsample if large) ----
n = X_scaled.shape[0]
if n > 4000:
    rng = np.random.RandomState(42)
    idx = rng.choice(n, 4000, replace=False)
    X_tsne = X_scaled[idx]; labels_tsne = labels_km[idx]
    print("\nRunning t-SNE on 4000-sample subset...")
else:
    X_tsne = X_scaled; labels_tsne = labels_km
tsne = TSNE(n_components=2, init='random', learning_rate='auto', random_state=42)
tsne_proj = tsne.fit_transform(X_tsne)
plt.figure(figsize=(8,6))
plt.scatter(tsne_proj[:,0], tsne_proj[:,1], c=labels_tsne, s=8, cmap='viridis', alpha=0.7)
plt.title("t-SNE projection of clusters"); plt.show()

# ---- Cluster profiling (numeric features only) ----
numeric_cols = df_results.select_dtypes(include=[np.number]).columns.tolist()
for lbl in ['cluster_kmeans','cluster_minibatch']:
    if lbl in numeric_cols: numeric_cols.remove(lbl)

cluster_profile = df_results.groupby('cluster_kmeans')[numeric_cols].mean().round(3)
print("\nCluster profile (means) — showing first 10 features:")
print(cluster_profile.iloc[:, :10].T)        # print first 10 features transposed for readability

# save full profile CSV
cluster_profile.to_csv(OUT_PROFILE)
print(f"\nSaved cluster profile -> {OUT_PROFILE}")

# ---- Summary prints ----
print("\nSummary:")
print(" - n_samples:", n)
print(" - n_features (used):", df_model.shape[1])
print(" - chosen k:", best_k)
print(" - clustered CSV:", OUT_CLUSTERED)
print(" - profile CSV:", OUT_PROFILE)
```

```
Loaded: CC GENERAL.csv shape: (8950, 18)

Columns:
 ['CUST_ID', 'BALANCE', 'BALANCE_FREQUENCY', 'PURCHASES', 'ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE', 'PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY', 'PURCHASES_INSTALLMENTS_

Missing values per column:
 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                         1
PAYMENTS                             0
MINIMUM_PAYMENTS                   313
PRC_FULL_PAYMENT                     0
TENURE                               0
dtype: int64
Dropping id-like columns: ['CUST_ID']

Preprocessing done. Numeric features used: 17
```


Elbow Method (Inertia)
Inertia
130000
120000
110000
100000
90000
80000
70000
2
3
4
5
6
7
8
9
10
k


Silhouette Scores
Silhouette Score
0.25
0.24
0.23
0.22
0.21
0.20
2
3
4
5
6
7
8
9
10
k


t-SNE projection of clusters
60
40
20
0
−20
−40
−60
−60
−40
−20
0
20
40
60


KMeans clusters (PCA-2) with centroids
PC2
PC1
25
20
15
10
5
0
0
5
10
15
20
25
30
2
0
1

```
Inertias: {2: 127784.53, 3: 111975.04, 4: 99061.94, 5: 91490.5, 6: 84826.59, 7: 79856.16, 8: 74484.88, 9: 69828.7, 10: 66466.41}
Silhouette scores: {2: np.float64(0.21), 3: np.float64(0.251), 4: np.float64(0.198), 5: np.float64(0.193), 6: np.float64(0.203), 7: np.float64(0.208), 8: np.float64(0.222), 9: np.float64(0.226), 10: np.float64(0.22)}
Chosen k (max silhouette): 3

Saved clustered data -> cc_general_with_clusters.csv

Cluster counts (KMeans):
cluster_kmeans
0    6119
1    1235
2    1596

Cluster counts (MiniBatchKMeans):
cluster_minibatch
0    4595
1    3577
2     778
```

```
Cluster profile (means) - showing first 10 features:
cluster_kmeans                          0         1         2
BALANCE                           799.751  2220.003  3989.142
BALANCE_FREQUENCY                   0.835     0.981     0.958
PURCHASES                         505.532  4268.521   384.528
ONEOFF_PURCHASES                  253.120  2717.829   248.718
INSTALLMENTS_PURCHASES            252.733  1551.178   135.888
CASH_ADVANCE                      330.818   458.421  3866.210
PURCHASES_FREQUENCY                 0.465     0.949     0.232
ONEOFF_PURCHASES_FREQUENCY          0.132     0.668     0.111
PURCHASES_INSTALLMENTS_FREQUENCY    0.345     0.745     0.144
CASH_ADVANCE_FREQUENCY              0.068     0.063     0.447

Saved cluster profile -> cluster_profile.csv

Summary:
 - n_samples: 8950
 - n_features (used): 17
 - chosen k: 3
 - clustered CSV: cc_general_with_clusters.csv
 - profile CSV: cluster_profile.csv
```