```
#!/usr/bin/env python3
import os, argparse
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import LabelEncoder

OUTPUT_DIR = "outputs"
DATA_PATH = os.path.join("data", "Mall_Customers.csv")

def preprocess(df):
    df = df.copy()
    if "Genre" in df.columns and df["Genre"].dtype == object:
        df["Genre"] = LabelEncoder().fit_transform(df["Genre"])
    if "CustomerID" in df.columns:
        df = df.drop(columns=["CustomerID"])
    for c in df.columns:
        df[c] = pd.to_numeric(df[c], errors="coerce")
    return df.dropna().reset_index(drop=True)

def elbow(X, kmin=2, kmax=10, path=None):
    w = []
    for k in range(kmin, kmax+1):
        km = KMeans(n_clusters=k, init="k-means++", random_state=42, n_init=10).fit(X)
        w.append(km.inertia_)
    if path:
        plt.figure()
```

```
        plt.plot(range(kmin, kmax+1), w, marker="o")
        plt.title("Elbow Method for Optimal k")
        plt.xlabel("Number of clusters (k)")
        plt.ylabel("WCSS (Inertia)")
        plt.tight_layout()
        plt.savefig(path, dpi=200)
        plt.close()
    return w

def main(args):
    os.makedirs(OUTPUT_DIR, exist_ok=True)
    df = pd.read_csv(args.data)
    dfp = preprocess(df)
    X = dfp.values

    elbow(X, 2, 10, os.path.join(OUTPUT_DIR, "elbow_method.png"))

    k = args.k if args.k is not None else 5
    km = KMeans(n_clusters=k, init="k-means++", random_state=42, n_init=10).fit(X)
    labels = km.labels_
    sil = silhouette_score(X, labels)

    with open(os.path.join(OUTPUT_DIR, "clustering_report.txt"), "w") as f:
        f.write(f"Chosen k: {k}\nSilhouette Score: {sil:.4f}\n")
        f.write("Cluster Centers:\n")
        for i, c in enumerate(km.cluster_centers_):
            f.write(f"{i}: {c.tolist()}\n")

    if X.shape[1] >= 2:
```

```
        plt.figure()
        plt.scatter(X[:,0], X[:,1], s=35, marker="o")
        plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1], marker="X", s=110)
        plt.title(f"K-Means Clusters (k={k}) - First 2 Features")
        plt.xlabel(dfp.columns[0])
        plt.ylabel(dfp.columns[1])
        plt.tight_layout()
        plt.savefig(os.path.join(OUTPUT_DIR, "clusters_first2.png"), dpi=200)
        plt.close()

if __name__ == "__main__":
    p = argparse.ArgumentParser()
    p.add_argument("--data", type=str, default=DATA_PATH)
    p.add_argument("--k", type=int, default=None)
    main(p.parse_args())
```