LAMPIRAN

Lampiran Perbandingan Standar Intensitas Konsumsi Energi (IKE) dengan Hasil Klasterisasi *K-Means*

Untuk melengkapi hasil klasterisasi pada Bab 4, berikut ini ditampilkan perbandingan antara hasil klaster *K-Means* (berdasarkan konsumsi energi harian dalam kWh/hari) dengan klasifikasi Standar Intensitas Konsumsi Energi (IKE) yang dihitung dalam kWh/m²/bulan. Delapan ruangan dipilih sebagai sampel agar mewakili variasi lantai, luas, dan jenis ruangan (ber-AC maupun non-AC).

No	Lantai	Ruangan	Luas (m²)	Jenis Ruangan	Rata-rata Konsumsi (kWh/hari)	Konsumsi Bulanan* (kWh/bulan)	IKE (kWh/m²/bulan)	Klasifikasi Standar IKE**	Hasil Klaster *K-Means* (Bab 4)
1	2	Kaprodi T. Mesin	28,8	Ber-AC	5,30	116,60	4,05	Sangat Efisien	*Cluster* 3 (Tinggi)
2	2	Admin T. Mesin	28,8	Non-AC	4,34	95,48	3,32	Boros	*Cluster* 3 (Tinggi)
3	3	Wakil Dekan Pertanian	17,28	Non-AC	1,70	37,40	2,16	Efisien	*Cluster* 1 (Rendah)
4	3	Dekan Fakultas Pertanian	37,8	Ber-AC	6,75	148,50	3,93	Sangat Efisien	*Cluster* 3 (Tinggi)
5	4	Dekan Fakultas Teknik	37,8	Ber-AC	5,55	122,10	3,23	Sangat Efisien	*Cluster* 3 (Tinggi)
6	4	Wakil Dekan Fakultas Teknik	17,28	Non-AC	1,70	37,40	2,16	Efisien	*Cluster* 1 (Rendah)
7	5	Admin Informatika	25,92	Non-AC	1,57	34,54	1,33	Sangat Efisien	*Cluster* 1 (Rendah)
8	5	Kaprodi Informatika	21,6	Ber-AC	5,30	116,60	5,40	Sangat Efisien	*Cluster* 3 (Tinggi)

** Konsumsi energi bulanan dihitung dengan asumsi 22 hari kerja per bulan (mengikuti pola operasional kampus 5 hari kerja per minggu, Senin–Jumat, seperti dijelaskan pada Bab 4), yaitu rata-rata konsumsi harian (kWh/hari) dikali 22.*

*** Klasifikasi ini mengacu pada Standar IKE Bangunan Gedung Indonesia yang disusun oleh Departemen Pendidikan Nasional RI (2004), sebagaimana dikutip dalam Saing, Krismahariyanto dan Fahdillah (2019), serta SNI 03-6196-2000 tentang Prosedur Audit Energi pada Bangunan Gedung (Badan Standardisasi Nasional, 2000) yang menjadi dasar Peraturan Menteri ESDM No. 13 Tahun 2012 tentang Penghematan Pemakaian Tenaga Listrik (Kementerian ESDM RI, 2012). Kriteria ruangan ber-AC: Sangat Efisien ≤ 7,92; Efisien 7,92–12,08; Cukup Efisien 12,08–14,58; Agak Boros 14,58–19,17; Boros 19,17–23,75; Sangat Boros 23,75–37,75 kWh/m²/bulan. Kriteria ruangan non-AC: Sangat Efisien ≤ 1,67; Efisien 1,67–2,5; Boros 2,5–3,34; Sangat Boros 3,34–4,17 kWh/m²/bulan.*

Dari tabel di atas terlihat bahwa hasil klaster *K-Means* tidak selalu sama dengan klasifikasi Standar IKE, dan hal ini wajar karena keduanya memang mengukur hal yang berbeda. Klaster *K-Means* dihitung dari besar konsumsi energi harian (kWh/hari) tanpa memperhitungkan luas ruangan, sedangkan Standar IKE menghitung efisiensi energi per luas ruangan (kWh/m²/bulan). Sebagai contoh, keempat ruangan ber-AC (Kaprodi T. Mesin, Dekan Fakultas Pertanian, Dekan Fakultas Teknik, dan Kaprodi Informatika) masuk *Cluster* 3 (Tinggi) karena konsumsi hariannya besar, tetapi karena ruangannya cukup luas, hasil IKE-nya justru Sangat Efisien. Sebaliknya, Admin T. Mesin yang juga masuk *Cluster* 3 (Tinggi) malah tergolong Boros menurut IKE karena ruangannya kecil dan tanpa AC. Jadi perbedaan hasil ini bukan berarti ada yang salah, melainkan menunjukkan bahwa konsumsi energi mutlak dan intensitas energi per luas adalah dua cara pandang yang saling melengkapi untuk menilai efisiensi energi tiap ruangan.

Lampiran Proses *Clustering K-Means*

Berikut disajikan hasil lengkap proses *clustering K-Means* terhadap 55 ruangan berdasarkan data konsumsi energi listrik harian (Senin–Jumat), mencakup nilai konsumsi tiap hari, rata-rata konsumsi harian, hasil penetapan cluster, dan kategori konsumsi (Rendah, Sedang, Tinggi) untuk setiap ruangan.

Lampiran 1. Hasil *Clustering K-Means* Seluruh Ruangan (55 Ruangan)

No	Lantai	Ruangan	Senin (kWh)	Selasa (kWh)	Rabu (kWh)	Kamis (kWh)	Jumat (kWh)	Rata-rata (kWh/hari)	Cluster	Kategori
1	1	*Cleaning Service*	0,13	0,13	0,13	0,13	0,13	0,13	*Cluster* 1	Rendah
2	1	B102	2,24	1,79	1,49	2,99	2,31	2,17	*Cluster* 2	Sedang
3	1	B105	2,69	3,36	3,14	4,48	0,00	2,73	*Cluster* 2	Sedang
4	1	B106	1,79	3,36	2,69	3,36	3,36	2,91	*Cluster* 2	Sedang
5	1	B103	0,00	2,69	3,14	2,02	3,36	2,24	*Cluster* 2	Sedang
6	1	B104	0,00	3,36	3,36	2,24	0,00	1,79	*Cluster* 1	Rendah
7	1	B101	0,00	2,24	3,73	2,54	4,48	2,60	*Cluster* 2	Sedang
8	2	Kaprodi T. Mesin	5,30	5,30	5,30	5,30	5,30	5,30	*Cluster* 3	Tinggi

No	Lantai	Ruangan	Senin (kWh)	Selasa (kWh)	Rabu (kWh)	Kamis (kWh)	Jumat (kWh)	Rata-rata (kWh/hari)	Cluster	Kategori
9	2	Admin T. Mesin	4,34	4,34	4,34	4,34	4,34	4,34	*Cluster* 3	Tinggi
10	2	Ruang Dosen T. Mesin	0,51	0,51	0,51	0,51	0,51	0,51	*Cluster* 1	Rendah
11	2	B201	3,73	4,48	2,99	2,99	5,15	3,87	*Cluster* 3	Tinggi
12	2	B202	3,73	3,73	5,23	2,99	3,73	3,88	*Cluster* 3	Tinggi
13	2	B203	2,24	4,48	3,73	3,73	5,97	4,03	*Cluster* 3	Tinggi
14	2	B204	1,49	4,48	4,48	4,48	3,73	3,73	*Cluster* 2	Sedang
15	2	B205	3,73	3,73	3,73	0,00	0,00	2,24	*Cluster* 1	Rendah
16	2	B206	3,73	3,73	3,73	3,73	2,24	3,43	*Cluster* 2	Sedang
17	2	B207	3,73	3,73	4,48	3,73	3,58	3,85	*Cluster* 3	Tinggi
18	2	B208	0,00	3,73	2,24	0,00	3,81	1,96	*Cluster* 2	Sedang
19	3	Wakil Dekan Pertanian	1,70	1,70	1,70	1,70	1,70	1,70	*Cluster* 1	Rendah
20	3	BAAK Fakultas Pertanian/Ruang Dosen Agro	4,34	4,34	4,34	4,34	4,34	4,34	*Cluster* 3	Tinggi
21	3	Dekan Fakultas Pertanian	6,75	6,75	6,75	6,75	6,75	6,75	*Cluster* 3	Tinggi

No	Lantai	Ruangan	Senin (kWh)	Selasa (kWh)	Rabu (kWh)	Kamis (kWh)	Jumat (kWh)	Rata-rata (kWh/hari)	Cluster	Kategori
22	3	Admin Fakultas Pertanian	1,70	1,70	1,70	1,70	1,70	1,70	*Cluster* 1	Rendah
23	3	Dosen Managemen	0,13	0,13	0,13	0,13	0,13	0,13	*Cluster* 1	Rendah
24	3	Dosen Agroteknologi	0,13	0,13	0,13	0,13	0,13	0,13	*Cluster* 1	Rendah
25	3	Kaprodi Agroteknologi	1,57	1,57	1,57	1,57	1,57	1,57	*Cluster* 1	Rendah
26	3	Admin Prodi Managemen	2,77	2,77	2,77	2,77	2,77	2,77	*Cluster* 2	Sedang
27	3	B301	2,24	2,99	1,49	0,75	0,75	1,64	*Cluster* 1	Rendah
28	3	B302	2,99	2,24	2,99	1,49	0,00	1,94	*Cluster* 1	Rendah
29	3	B305	2,39	3,36	2,91	3,58	2,99	3,05	*Cluster* 2	Sedang
30	3	B304	2,24	4,48	1,12	2,24	2,24	2,46	*Cluster* 2	Sedang
31	3	B306	2,24	3,36	4,48	2,91	3,36	3,27	*Cluster* 2	Sedang
32	3	B307	1,79	3,36	1,87	3,36	3,96	2,87	*Cluster* 2	Sedang
33	3	B303	0,00	0,75	0,00	0,00	1,49	0,45	*Cluster* 1	Rendah
34	4	Klinik Kesehatan	1,57	1,57	1,57	1,57	1,57	1,57	*Cluster* 1	Rendah
35	4	Admin Elektro	4,34	4,34	4,34	4,34	4,34	4,34	*Cluster*	Tinggi

No	Lantai	Ruangan	Senin (kWh)	Selasa (kWh)	Rabu (kWh)	Kamis (kWh)	Jumat (kWh)	Rata-rata (kWh/hari)	Cluster	Kategori
									3	
36	4	Dosen Elektro	0,13	0,13	0,13	0,13	0,13	0,13	*Cluster* 1	Rendah
37	4	Ruang Server	1,57	1,57	1,57	1,57	1,57	1,57	*Cluster* 1	Rendah
38	4	Kaprodi Elektro	1,57	1,57	1,57	1,57	1,57	1,57	*Cluster* 1	Rendah
39	4	Dekan Fakultas Teknik	5,55	5,55	5,55	5,55	5,55	5,55	*Cluster* 3	Tinggi
40	4	Wakil Dekan Fakultas Teknik	1,70	1,70	1,70	1,70	1,70	1,70	*Cluster* 1	Rendah
41	4	Admin Fakultas Teknik	4,34	4,34	4,34	4,34	4,34	4,34	*Cluster* 3	Tinggi
42	4	B401	2,99	2,87	2,99	2,24	2,61	2,74	*Cluster* 2	Sedang
43	4	B402	3,73	2,99	3,73	2,99	2,24	3,14	*Cluster* 2	Sedang
44	4	B403	3,85	4,07	3,66	4,11	2,24	3,58	*Cluster* 2	Sedang
45	4	B404	3,29	3,58	3,55	3,55	2,87	3,37	*Cluster* 2	Sedang
46	4	B406	1,79	3,55	2,87	3,58	3,21	3,00	*Cluster* 2	Sedang
47	4	B405	1,42	3,62	2,50	3,58	2,84	2,79	*Cluster* 2	Sedang
48	4	B407	1,79	3,58	3,58	2,87	2,87	2,94	*Cluster* 2	Sedang

No	Lantai	Ruangan	Senin (kWh)	Selasa (kWh)	Rabu (kWh)	Kamis (kWh)	Jumat (kWh)	Rata-rata (kWh/hari)	Cluster	Kategori
49	5	Dosen Informatika	1,70	1,70	1,70	1,70	1,70	1,70	*Cluster* 1	Rendah
50	5	Dosen Informatika 2	0,26	0,26	0,26	0,26	0,26	0,26	*Cluster* 1	Rendah
51	5	Admin Informatika	1,57	1,57	1,57	1,57	1,57	1,57	*Cluster* 1	Rendah
52	5	Kaprodi Informatika	5,30	5,30	5,30	5,30	5,30	5,30	*Cluster* 3	Tinggi
53	5	B501	2,54	3,25	2,87	2,50	2,54	2,74	*Cluster* 2	Sedang
54	5	B503	2,87	1,42	3,58	2,17	3,25	2,66	*Cluster* 2	Sedang
55	5	B502	0,75	3,58	3,58	3,58	2,80	2,86	*Cluster* 2	Sedang

Lampiran Koding

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score

from openpyxl.drawing.image import Image as XLImage
from openpyxl.styles import Font, PatternFill, Alignment, Border, Side
from openpyxl.utils import get_column_letter

# ------------------------------------------------------------------------
# 0. KONFIGURASI
# ------------------------------------------------------------------------
# Data TIDAK distandarisasi (Z-score).
# K-Means dijalankan langsung pada nilai kWh mentah (Senin-Jumat) karena
# kelima atribut sudah dalam satuan yang sama (kWh/hari).
INPUT_FILE   = "DataKmeans.xlsx"      # file data mentah
OUTPUT_FILE  = "Hasil_Clustering_Energi.xlsx"
K_RANGE      = range(1, 9)              # menguji K=1 s.d. K=8
RANDOM_STATE = 42                       # agar hasil selalu sama tiap dijalankan
MAX_ITER     = 500                      # sesuai BAB III (maks. 500 iterasi)
N_INIT       = 10                       # 10 kali restart, ambil WCSS terkecil
AMBANG_PERSEN = 25.0                    # ambang batas penurunan WCSS (BAB IV)
FITUR_HARIAN = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat"]
```

```
plt.rcParams["figure.dpi"] = 120

# ------------------------------------------------------------------------
# 1. MEMBACA DAN MEMBERSIHKAN DATA
# ------------------------------------------------------------------------
def load_data(path: str) -> pd.DataFrame:
    df = pd.read_excel(path)
    df.columns = [c.strip() for c in df.columns]

    # pastikan tidak ada data kosong pada kolom numerik yang dipakai clustering
    kolom_numerik = FITUR_HARIAN + ["Rata-rata (kWh/hari)"]
    df[kolom_numerik] = df[kolom_numerik].fillna(0)

    return df


# ------------------------------------------------------------------------
# 2. MENENTUKAN JUMLAH CLUSTER OPTIMAL (ELBOW METHOD - % PENURUNAN WCSS)
# ------------------------------------------------------------------------
# K dipilih berdasarkan titik di mana persentase
# penurunan WCSS mulai melambat signifikan (di bawah ~25%), BUKAN dari
# Silhouette Score. Silhouette Score tetap dihitung & dilaporkan sebagai
# metrik pembanding/validasi tambahan.
def cari_k_optimal(X: np.ndarray, k_range) -> pd.DataFrame:
    hasil = []
    for k in k_range:
        km = KMeans(n_clusters=k, init="K-Means++", random_state=RANDOM_STATE,
                    n_init=N_INIT, max_iter=MAX_ITER)
        km.fit(X)
```

```
        WCSS = km.inertia_                          # Within-Cluster Sum of Squares

        if 1 < k < len(X):
            sil = silhouette_score(X, km.labels_)
        else:
            sil = np.nan                             # silhouette tidak terdefinisi untuk k=1 atau k=n

        hasil.append({
            "k": k,
            "WCSS": WCSS,
            "Silhouette Score": sil,
            "Jumlah Iterasi Terbaik (n_iter_)": km.n_iter_,
        })

    df_eval = pd.DataFrame(hasil)
    df_eval["Penurunan WCSS (%)"] = (
        -df_eval["WCSS"].pct_change() * 100
    ).round(2)
    df_eval["Silhouette Score"] = df_eval["Silhouette Score"].round(4)
    df_eval["WCSS"] = df_eval["WCSS"].round(4)
    kolom_urut = ["k", "WCSS", "Penurunan WCSS (%)", "Silhouette Score",
                  "Jumlah Iterasi Terbaik (n_iter_)"]
    return df_eval[kolom_urut]


def plot_Elbow(df_eval: pd.DataFrame, out_path: str, k_pilihan: int):
    fig, ax1 = plt.subplots(figsize=(6.5, 4.5))
    ax1.plot(df_eval["k"], df_eval["WCSS"], marker="o", color="#2563eb", label="WCSS")
    ax1.axvline(k_pilihan, color="#ef4444", linestyle="--", label=f"K optimal = {k_pilihan}")
```

```
    ax1.set_xlabel("Jumlah Cluster (K)")
    ax1.set_ylabel("WCSS", color="#2563eb")
    ax1.grid(alpha=0.3)

    ax2 = ax1.twinx()
    ax2.plot(df_eval["k"], df_eval["Silhouette Score"], marker="s",
             color="#16a34a", linestyle=":", label="Silhouette Score")
    ax2.set_ylabel("Silhouette Score", color="#16a34a")

    lines1, labels1 = ax1.get_legend_handles_labels()
    lines2, labels2 = ax2.get_legend_handles_labels()
    ax1.legend(lines1 + lines2, labels1 + labels2, loc="upper right", fontsize=8)

    plt.title("Elbow Method (WCSS) & Silhouette Score vs Jumlah Cluster")
    fig.tight_layout()
    fig.savefig(out_path, bbox_inches="tight")
    plt.close(fig)


# ---------------------------------------------------------------------------
# 3. DETAIL PROSES K-MEANS: PER-PERCOBAAN (n_init) & PER-ITERASI
# ---------------------------------------------------------------------------
def rekam_detail_percobaan(X: np.ndarray, k_final: int) -> pd.DataFrame:
    """
    Menjalankan K-Means sebanyak N_INIT kali secara terpisah (masing-masing
    dengan inisialisasi K-Means++ yang berbeda / seed berbeda) untuk
    memperlihatkan WCSS akhir & jumlah iterasi tiap percobaan, sebelum
    akhirnya dipilih percobaan dengan WCSS terkecil sebagai model final
    (sesuai parameter n_init=10 pada model resmi).
```

```
    """
    hasil = []
    for percobaan in range(1, N_INIT + 1):
        seed_percobaan = RANDOM_STATE + percobaan  # seed berbeda tiap percobaan
        km = KMeans(n_clusters=k_final, init="K-Means++", random_state=seed_percobaan,
                    n_init=1, max_iter=MAX_ITER)
        km.fit(X)
        hasil.append({
            "Percobaan ke-": percobaan,
            "Seed Inisialisasi": seed_percobaan,
            "WCSS Akhir": round(km.inertia_, 4),
            "Jumlah Iterasi sampai Konvergen": km.n_iter_,
        })

    df_percobaan = pd.DataFrame(hasil)
    WCSS_terbaik = df_percobaan["WCSS Akhir"].min()
    df_percobaan["Terpilih sebagai Model Final"] = np.where(
        df_percobaan["WCSS Akhir"] == WCSS_terbaik, "Ya", ""
    )
    return df_percobaan


def rekam_riwayat_iterasi(X: np.ndarray, k_final: int) -> pd.DataFrame:
    """
    Merekam nilai WCSS pada SETIAP iterasi algoritma Lloyd (assignment ->
    update centroid) untuk satu proses K-Means (inisialisasi K-Means++
    dengan random_state=RANDOM_STATE, n_init=1), sampai konvergen.

    Caranya: menjalankan ulang K-Means dengan batas max_iter yang bertambah
```

```
    1 iterasi setiap kali (mulai dari titik awal yang identik karena
    random_state tetap sama), sehingga nilai n_iter_ dan inertia_ pada tiap
    langkah mencerminkan kondisi asli algoritma pada iterasi tersebut.
    """
    riwayat = []
    WCSS_sebelumnya = None
    for batas_iterasi in range(1, MAX_ITER + 1):
        km = KMeans(n_clusters=k_final, init="K-Means++", random_state=RANDOM_STATE,
                    n_init=1, max_iter=batas_iterasi, algorithm="lloyd")
        km.fit(X)
        WCSS = km.inertia_
        perubahan_WCSS = None if WCSS_sebelumnya is None else round(WCSS_sebelumnya - WCSS, 6)
        riwayat.append({
            "Iterasi ke-": batas_iterasi,
            "WCSS": round(WCSS, 4),
            "Penurunan WCSS dari Iterasi Sebelumnya": perubahan_WCSS,
            "Status": "Konvergen" if km.n_iter_ < batas_iterasi else "Berjalan",
        })
        WCSS_sebelumnya = WCSS
        if km.n_iter_ < batas_iterasi:
            break  # sudah konvergen sebelum mencapai batas_iterasi -> berhenti mencatat

    return pd.DataFrame(riwayat)


# ------------------------------------------------------------------------
# 4. K-MEANS CLUSTERING FINAL
# ------------------------------------------------------------------------
def jalankan_kmeans(X: np.ndarray, k_final: int):
```

```
    # Inisialisasi standar: K-Means++ dengan multiple restart (n_init=10),
    # dipilih hasil dengan WCSS terkecil dari 10 percobaan.
    km = KMeans(n_clusters=k_final, init="K-Means++", random_state=RANDOM_STATE,
                n_init=N_INIT, max_iter=MAX_ITER)
    label = km.fit_predict(X)
    return km, label


def buat_tabel_centroid(km: KMeans, mapping: dict) -> pd.DataFrame:
    df_centroid = pd.DataFrame(km.cluster_centers_, columns=FITUR_HARIAN).round(3)
    df_centroid.insert(0, "Cluster", range(len(df_centroid)))
    df_centroid["Kategori"] = df_centroid["Cluster"].map(mapping)
    df_centroid["Rata-rata Centroid (kWh/hari)"] = df_centroid[FITUR_HARIAN].mean(axis=1).round(3)
    return df_centroid


def hitung_jarak_ke_centroid(X: np.ndarray, km: KMeans, label: np.ndarray) -> np.ndarray:
    centroid_titik = km.cluster_centers_[label]
    jarak = np.sqrt(np.sum((X - centroid_titik) ** 2, axis=1))
    return np.round(jarak, 4)


def buat_ringkasan_per_cluster(df: pd.DataFrame, km: KMeans, mapping: dict,
                               kategori_urut: list) -> pd.DataFrame:
    """
    Ringkasan proses untuk tiap cluster hasil model final: jumlah anggota,
    kontribusi tiap cluster terhadap WCSS total, serta jarak anggota ke
    centroid clusternya (rata-rata & maksimum).
    """
```

```
    WCSS_total = km.inertia_
    baris = []
    for cl in sorted(mapping.keys()):
        anggota = df[df["Cluster"] == cl]
        WCSS_cluster = float(np.sum(anggota["Jarak ke Centroid"].values ** 2))
        baris.append({
            "Cluster": cl,
            "Kategori": mapping[cl],
            "Jumlah Anggota (Ruangan)": len(anggota),
            "Persentase Anggota (%)": round(len(anggota) / len(df) * 100, 2),
            "Rata-rata kWh/hari": round(anggota["Rata-rata (kWh/hari)"].mean(), 3),
            "Min kWh/hari": round(anggota["Rata-rata (kWh/hari)"].min(), 3),
            "Max kWh/hari": round(anggota["Rata-rata (kWh/hari)"].max(), 3),
            "WCSS Cluster": round(WCSS_cluster, 4),
            "Kontribusi thd WCSS Total (%)": round(WCSS_cluster / WCSS_total * 100, 2) if WCSS_total else 0,
            "Rata-rata Jarak ke Centroid": round(anggota["Jarak ke Centroid"].mean(), 4),
            "Jarak Maksimum ke Centroid": round(anggota["Jarak ke Centroid"].max(), 4),
        })

    df_ringkasan = pd.DataFrame(baris)
    urutan_map = {kat: i for i, kat in enumerate(kategori_urut)}
    df_ringkasan["_urut"] = df_ringkasan["Kategori"].map(urutan_map)
    df_ringkasan = df_ringkasan.sort_values("_urut").drop(columns="_urut").reset_index(drop=True)
    return df_ringkasan


def beri_label_kategori(df: pd.DataFrame, label_col="Cluster", nilai_col="Rata-rata (kWh/hari)"):
    """
    Mengubah nomor cluster (0,1,2,...) yang urutannya acak menjadi label
```

```
    kategori yang bermakna: Rendah / Sedang / Tinggi, berdasarkan urutan
    rata-rata konsumsi energi tiap cluster (cluster dgn rata-rata terkecil = Rendah).
    """
    urutan_cluster = (
        df.groupby(label_col)[nilai_col].mean().sort_values().index.tolist()
    )
    k = len(urutan_cluster)

    if k == 3:
        nama_kategori = ["Rendah", "Sedang", "Tinggi"]
    else:
        nama_kategori = [f"Kategori {i + 1}" for i in range(k)]

    mapping = {cl: nama for cl, nama in zip(urutan_cluster, nama_kategori)}
    df["Kategori Konsumsi"] = df[label_col].map(mapping)
    return df, mapping


# ---------------------------------------------------------------------------
# 5. VISUALISASI HASIL AKHIR
# ---------------------------------------------------------------------------
def buat_palet_warna(kategori_urut: list) -> dict:
    """
    Membuat palet warna hijau->merah (rendah->tinggi) yang otomatis
    menyesuaikan jumlah kategori, bukan hanya untuk K=3 (Rendah/Sedang/Tinggi).
    """
    if kategori_urut == ["Rendah", "Sedang", "Tinggi"]:
        return {"Rendah": "#22c55e", "Sedang": "#eab308", "Tinggi": "#ef4444"}
    cmap = plt.get_cmap("RdYlGn_r")
```

```
    n = max(len(kategori_urut) - 1, 1)
    return {kat: cmap(i / n) for i, kat in enumerate(kategori_urut)}


def plot_pca_scatter(X: np.ndarray, df: pd.DataFrame, out_path: str, kategori_urut: list):
    pca = PCA(n_components=2, random_state=RANDOM_STATE)
    komponen = pca.fit_transform(X)
    df["PCA1"], df["PCA2"] = komponen[:, 0], komponen[:, 1]

    warna = buat_palet_warna(kategori_urut)
    plt.figure(figsize=(7, 5.5))
    for kategori, sub in df.groupby("Kategori Konsumsi"):
        plt.scatter(
            sub["PCA1"], sub["PCA2"],
            label=kategori,
            s=70, alpha=0.85,
            color=warna.get(kategori, None),
            edgecolor="white", linewidth=0.5,
        )
    for _, row in df.iterrows():
        plt.annotate(row["Ruangan"], (row["PCA1"], row["PCA2"]),
                     fontsize=6, alpha=0.7, xytext=(3, 3), textcoords="offset points")

    var_explained = pca.explained_variance_ratio_ * 100
    plt.xlabel(f"PC1 ({var_explained[0]:.1f}% variansi)")
    plt.ylabel(f"PC2 ({var_explained[1]:.1f}% variansi)")
    plt.title("Visualisasi Cluster Konsumsi Energi per Ruangan (PCA)")
    plt.legend(title="Kategori")
    plt.grid(alpha=0.25)
```

```
    plt.tight_layout()
    plt.savefig(out_path, bbox_inches="tight")
    plt.close()
    return df


def plot_bar_rata_rata(df: pd.DataFrame, out_path: str, kategori_urut: list):
    urut = df.sort_values("Rata-rata (kWh/hari)")
    palet = buat_palet_warna(kategori_urut)
    warna = urut["Kategori Konsumsi"].map(palet)
    plt.figure(figsize=(9, 12))
    plt.barh(urut["Ruangan"], urut["Rata-rata (kWh/hari)"], color=warna)
    plt.xlabel("Rata-rata Konsumsi Energi (kWh/hari)")
    plt.title("Rata-rata Konsumsi Energi Listrik per Ruangan berdasarkan Cluster")
    plt.tight_layout()
    plt.savefig(out_path, bbox_inches="tight")
    plt.close()


# ---------------------------------------------------------------------------
# 6. PENULISAN EXCEL (FORMAT RAPI: HEADER, LEBAR KOLOM, FREEZE PANE)
# ---------------------------------------------------------------------------
def format_sheet(ws, df: pd.DataFrame, freeze="A2"):
    """Menerapkan format header, lebar kolom otomatis, dan freeze pane."""
    for col_idx, kolom in enumerate(df.columns, start=1):
        sel = ws.cell(row=1, column=col_idx)
        sel.font = FONT_HEADER
        sel.fill = FILL_HEADER
        sel.alignment = ALIGN_TENGAH
```

```
        sel.border = BORDER_TIPIS

        # lebar kolom menyesuaikan panjang konten terpanjang
        panjang_maks = max(
            [len(str(kolom))] + [len(str(v)) for v in df[kolom].astype(str).values]
        )
        ws.column_dimensions[get_column_letter(col_idx)].width = min(panjang_maks + 3, 45)

    for row in ws.iter_rows(min_row=2, max_row=ws.max_row, max_col=ws.max_column):
        for cell in row:
            cell.border = BORDER_TIPIS

    ws.freeze_panes = freeze


def tulis_sheet_gambar(writer, sheet_name: str, gambar_paths: list):
    """Membuat sheet khusus berisi grafik (PNG) yang ditempel langsung di Excel."""
    ws = writer.book.create_sheet(sheet_name)
    baris = 1
    for judul, path in gambar_paths:
        ws.cell(row=baris, column=1, value=judul).font = Font(bold=True, size=12)
        img = XLImage(path)
        img.anchor = f"A{baris + 1}"
        ws.add_image(img)
        baris += 30  # jarak antar grafik


# ---------------------------------------------------------------------------
# 7. PROGRAM UTAMA
```

```
# ------------------------------------------------------------------------
def main():
    # --- 1. Baca data ---
    df_mentah = load_data(INPUT_FILE)
    df = df_mentah.copy()
    print(f"Jumlah ruangan (data): {len(df)}")

    # --- 2. Fitur clustering: nilai kWh mentah, TANPA standarisasi ---
    X = df[FITUR_HARIAN].values.astype(float)

    # --- 3. Statistik deskriptif ---
    df_statistik = df[FITUR_HARIAN + ["Rata-rata (kWh/hari)"]].describe().round(3)
    df_statistik.insert(0, "Statistik", df_statistik.index)
    df_statistik = df_statistik.reset_index(drop=True)

    # --- 4. Cari k optimal (Elbow Method: K=1 s.d. K=8) ---
    df_eval = cari_k_optimal(X, K_RANGE)
    print("\nEvaluasi jumlah cluster (WCSS, % penurunan, Silhouette):")
    print(df_eval.to_string(index=False))

    kandidat = df_eval[df_eval["Penurunan WCSS (%)"] >= AMBANG_PERSEN]
    K_FINAL = int(kandidat["k"].max()) if not kandidat.empty else 3
    plot_Elbow(df_eval, "Elbow_method.png", K_FINAL)
    print(f"\n>> K optimal terpilih (penurunan WCSS masih >= {AMBANG_PERSEN}%): K={K_FINAL}")

    # --- 5. Rekam detail proses (n_init & riwayat iterasi) untuk K_FINAL ---
    df_percobaan = rekam_detail_percobaan(X, K_FINAL)
    print("\nDetail 10 percobaan inisialisasi (n_init):")
```

```
print(df_percobaan.to_string(index=False))

df_riwayat_iterasi = rekam_riwayat_iterasi(X, K_FINAL)
print(f"\nRiwayat iterasi K-Means (K={K_FINAL}) sampai konvergen "
      f"({len(df_riwayat_iterasi)} iterasi):")
print(df_riwayat_iterasi.to_string(index=False))

# --- 6. Jalankan K-Means final (model resmi, n_init=10) ---
km, label = jalankan_kmeans(X, K_FINAL)
df["Cluster"] = label
print(f"\nK-Means final dengan K={K_FINAL} -> WCSS = {km.inertia_:.4f}, "
      f"iterasi hingga konvergen = {km.n_iter_}")

# --- 7. Beri label kategori Rendah/Sedang/Tinggi ---
df, mapping = beri_label_kategori(df)
print("\nPemetaan cluster -> kategori:", mapping)

# --- 8. Jarak tiap titik ke centroid clusternya & tabel centroid ---
df["Jarak ke Centroid"] = hitung_jarak_ke_centroid(X, km, label)
df_centroid = buat_tabel_centroid(km, mapping)

# --- 8b. Ringkasan proses: info umum + rincian per cluster ---
kategori_urut_sementara = (
    ["Rendah", "Sedang", "Tinggi"] if K_FINAL == 3
    else sorted(set(mapping.values()), key=lambda x: int(x.split()[-1]))
)
df_ringkasan_cluster = buat_ringkasan_per_cluster(df, km, mapping, kategori_urut_sementara)
```

```
silhouette_final = round(silhouette_score(X, label), 4) if 1 < K_FINAL < len(X) else "N/A"
df_info_proses = pd.DataFrame([
    {"Info": "Jumlah Cluster (K) Terpilih", "Nilai": K_FINAL},
    {"Info": "Jumlah Ruangan (Total Data)", "Nilai": len(df)},
    {"Info": "Jumlah Iterasi Model Final sampai Konvergen", "Nilai": km.n_iter_},
    {"Info": "Jumlah Restart Inisialisasi (n_init)", "Nilai": N_INIT},
    {"Info": "Batas Maksimum Iterasi per Restart (max_iter)", "Nilai": MAX_ITER},
    {"Info": "Rata-rata Iterasi dari 10 Percobaan (n_init)",
     "Nilai": round(df_percobaan["Jumlah Iterasi sampai Konvergen"].mean(), 2)},
    {"Info": "Iterasi Tercepat antar 10 Percobaan",
     "Nilai": int(df_percobaan["Jumlah Iterasi sampai Konvergen"].min())},
    {"Info": "Iterasi Terlama antar 10 Percobaan",
     "Nilai": int(df_percobaan["Jumlah Iterasi sampai Konvergen"].max())},
    {"Info": "WCSS Model Final", "Nilai": round(km.inertia_, 4)},
    {"Info": "Silhouette Score Model Final", "Nilai": silhouette_final},
])

# --- 9. Ringkasan per kategori ---
# urutkan sesuai skala Rendah->Tinggi bila K=3, selain itu urut alami (Kategori 1..K)
kategori_urut = kategori_urut_sementara

ringkasan = (
    df.groupby("Kategori Konsumsi")
    .agg(
        Jumlah_Ruangan=("Ruangan", "count"),
        Rata_rata_kWh=("Rata-rata (kWh/hari)", "mean"),
        Min_kWh=("Rata-rata (kWh/hari)", "min"),
        Max_kWh=("Rata-rata (kWh/hari)", "max"),
```

```
            Std_kWh=("Rata-rata (kWh/hari)", "std"),
        )
        .reindex(kategori_urut)
        .round(3)
        .reset_index()
    )
    print("\nRingkasan per kategori:\n", ringkasan)

    ringkasan_lantai = (
        df.groupby(["Lantai", "Kategori Konsumsi"])
        .size()
        .unstack(fill_value=0)
        .reindex(columns=kategori_urut, fill_value=0)
        .reset_index()
    )
    print("\nDistribusi kategori per lantai:\n", ringkasan_lantai)

    # --- 10. Parameter & metadata ---
    df_parameter = pd.DataFrame([
        {"Parameter": "Jumlah data (ruangan)", "Nilai": len(df)},
        {"Parameter": "Fitur yang digunakan", "Nilai": ", ".join(FITUR_HARIAN)},
        {"Parameter": "Standarisasi (Z-score)", "Nilai": "Tidak (data kWh mentah)"},
        {"Parameter": "Rentang K yang diuji", "Nilai": f"{K_RANGE.start} s.d. {K_RANGE.stop - 1}"},
        {"Parameter": "Metode pemilihan K", "Nilai": f"Elbow Method (ambang penurunan WCSS >=
{AMBANG_PERSEN}%)"},
        {"Parameter": "K optimal terpilih", "Nilai": K_FINAL},
        {"Parameter": "Inisialisasi centroid", "Nilai": "K-Means++"},
        {"Parameter": "Jumlah restart (n_init)", "Nilai": N_INIT},
```

```
    {"Parameter": "Maksimum iterasi (max_iter)", "Nilai": MAX_ITER},
    {"Parameter": "Random state", "Nilai": RANDOM_STATE},
    {"Parameter": "WCSS model final", "Nilai": round(km.inertia_, 4)},
    {"Parameter": "Jumlah iterasi model final (n_iter_)", "Nilai": km.n_iter_},
    {"Parameter": "Silhouette Score model final",
     "Nilai": round(silhouette_score(X, label), 4) if 1 < K_FINAL < len(X) else "N/A"},
])

# --- 11. Visualisasi ---
df = plot_pca_scatter(X, df, "pca_scatter.png", kategori_urut)
plot_bar_rata_rata(df, "bar_rata_rata.png", kategori_urut)

# --- 12. Simpan seluruh hasil ke satu file Excel (multi-sheet + grafik) ---
kolom_output = [
    "No", "Lantai", "Ruangan", "Senin", "Selasa", "Rabu", "Kamis", "Jumat",
    "Rata-rata (kWh/hari)", "Cluster", "Kategori Konsumsi", "Jarak ke Centroid",
]

with pd.ExcelWriter(OUTPUT_FILE, engine="openpyxl") as writer:
    lembar = {
        "Parameter": df_parameter,
        "Data Mentah": df_mentah,
        "Statistik Deskriptif": df_statistik,
        "Evaluasi K (Elbow)": df_eval,
        "Detail Percobaan n_init": df_percobaan,
        "Riwayat Iterasi": df_riwayat_iterasi,
        "Info Proses Clustering": df_info_proses,
        "Ringkasan per Cluster": df_ringkasan_cluster,
```

```
            "Centroid Akhir": df_centroid,
            "Hasil Clustering": df[kolom_output],
            "Ringkasan Kategori": ringkasan,
            "Distribusi per Lantai": ringkasan_lantai,
        }

        for nama_sheet, data in lembar.items():
            data.to_excel(writer, sheet_name=nama_sheet, index=False)
            format_sheet(writer.sheets[nama_sheet], data)

        # sheet khusus grafik (gambar ditempel langsung, bukan hanya file terpisah)
        tulis_sheet_gambar(writer, "Grafik", [
            ("Elbow Method & Silhouette Score", "Elbow_method.png"),
            ("PCA Scatter - Visualisasi Cluster", "pca_scatter.png"),
            ("Rata-rata Konsumsi per Ruangan", "bar_rata_rata.png"),
        ])

    print("\nInfo proses clustering:\n", df_info_proses.to_string(index=False))
    print("\nRingkasan per cluster:\n", df_ringkasan_cluster.to_string(index=False))

    print(f"\nSelesai. Hasil lengkap disimpan di: {OUTPUT_FILE}")
    print("Sheet:", ", ".join(list(lembar.keys()) + ["Grafik"]))
    print("Grafik (file terpisah): Elbow_method.png, pca_scatter.png, bar_rata_rata.png")


if __name__ == "__main__":
    main()
```