

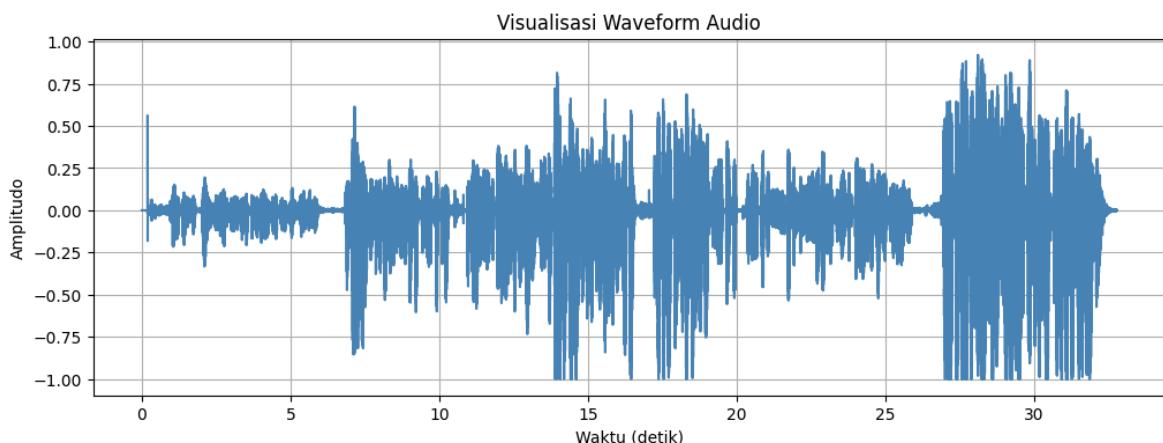
Nama : Muhammad Fakhri Nur NIM : 122140034 Link repositori :
<https://github.com/Sovenable/Worksheet-Multimedia.git>

Soal 1: Rekaman dan Analisis Suara Multi-Level
• Rekamlah suara Anda sendiri selama 25 detik dimana Anda membaca sebuah teks berita.
• Dalam 25 detik rekaman tersebut, Anda harus merekam:
– 5 detik pertama: suara sangat pelan dan berbisik
– 5 detik kedua: suara normal
– 5 detik ketiga: suara keras
– 5 detik keempat: suara cempreng (dibuat-buat cempreng)
– 5 detik terakhir: suara berteriak
• Rekam dalam format WAV (atau konversikan ke WAV sebelum dimuat ke notebook).
• Visualisasikan waveform dan spektrogram dari rekaman suara Anda.
• Sertakan penjelasan singkat mengenai hasil visualisasi tersebut.
• Lakukan resampling pada file audio Anda kemudian bandingkan kualitas dan durasinya.

```
In [3]: import librosa
import soundfile as sf
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import Audio
import os
file_path = 'soal 1.wav' # ganti dengan nama file kamu
y, sr = librosa.load(file_path, sr=None, mono=True) # sr=None agar pake sample
duration = len(y) / sr
print(f"Sample rate: {sr} Hz")
print(f"Durasi: {duration:.3f} s")
```

Sample rate: 48000 Hz
Durasi: 32.760 s

```
In [4]: time = np.linspace(0, duration, len(y)) # waktu (sumbu x)
plt.figure(figsize=(12, 4))
plt.plot(time, y, color='steelblue')
plt.title("Visualisasi Waveform Audio")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.show()
```

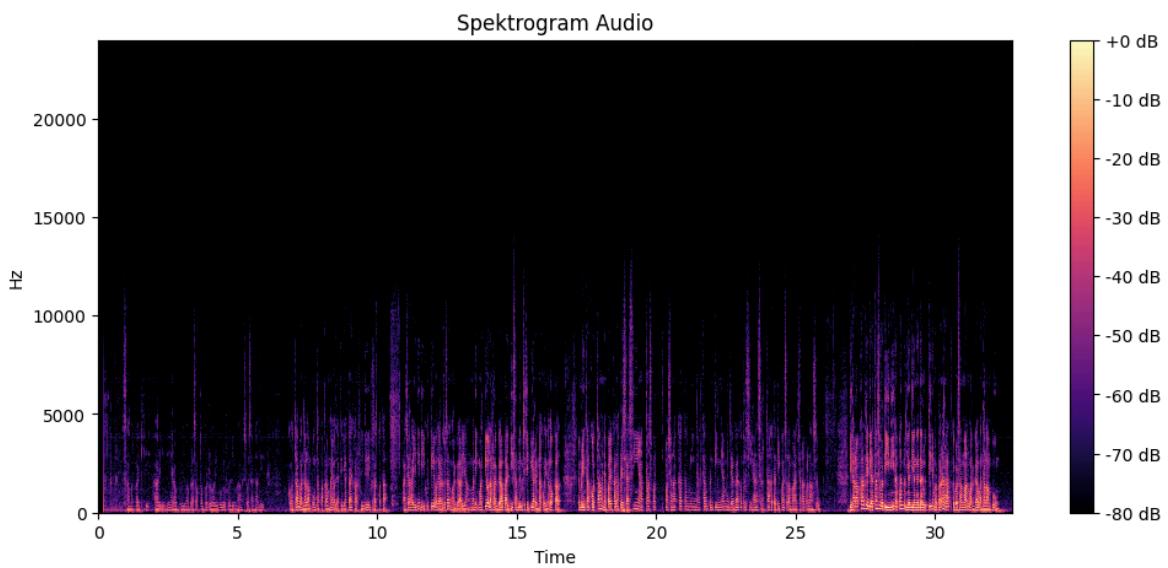


Dari hasil Wavefrom ini bisa dilihat di detik 0 sampai 5 amplitudo nya kecil yang menandakan suara pelan. di detik 5 sampai detik 12 mulai naik amplitudo nya yang menandakan itu suara normal dan bisa dilihat di detik 11 itu suara tarikan nafas saya. dari

detik 12 sampai 19 amplitudo nya naik lagi yang menandakan suara nya keras. kemudian di detik 20 sampai 27 dilihat dari amplitudo nya mirip dengan suara normal tapi kurang rapat suaranya, yang menandakan suara cempreng. dan dari detik 28 sampai akhir itu amplitudo nya udah full yang menandakan suara teriak

```
In [ ]: D = np.abs(librosa.stft(y, n_fft=2048, hop_length=512))
DB = librosa.amplitude_to_db(D, ref=np.max)

plt.figure(figsize=(12, 5))
librosa.display.specshow(DB, sr=sr, hop_length=512, x_axis='time', y_axis='hz',
plt.colorbar(format='%+2.0f dB')
plt.title('Spektrogram Audio')
plt.show()
```



Dari hasil Spektrogram ini, bisa dilihat di rata rata suara di bawah frekuensi 5000hz yang menandakan suara manusia dan warna warna kuning itu suara yang terdengar kuat, warna ungu ke hitam menunjukkan suara yang lebih rendah

```
In [ ]: # === 2 Resampling ke 16 kHz ===
target_sr = 16000
y_rs = librosa.resample(y, orig_sr=sr, target_sr=target_sr)
dur_rs = len(y_rs) / target_sr
print(f"\nAudio Setelah Resampling: Sample rate = {target_sr} Hz | Durasi = {dur_rs}\n")

# 🔍 Audio Hasil Resampling
print("🎧 Audio Hasil Resampling (16 kHz):")
display(ipd.Audio(y_rs, rate=target_sr))

# === 3 Visualisasi Waveform Perbandingan ===
plt.figure(figsize=(14,6))

plt.subplot(2,1,1)
librosa.display.waveshow(y, sr=sr)
plt.title(f"Waveform Asli ({sr} Hz)")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

plt.subplot(2,1,2)
librosa.display.waveshow(y_rs, sr=target_sr, color='orange')
plt.title(f"Waveform Setelah Resampling ({target_sr} Hz)")
```

```
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")

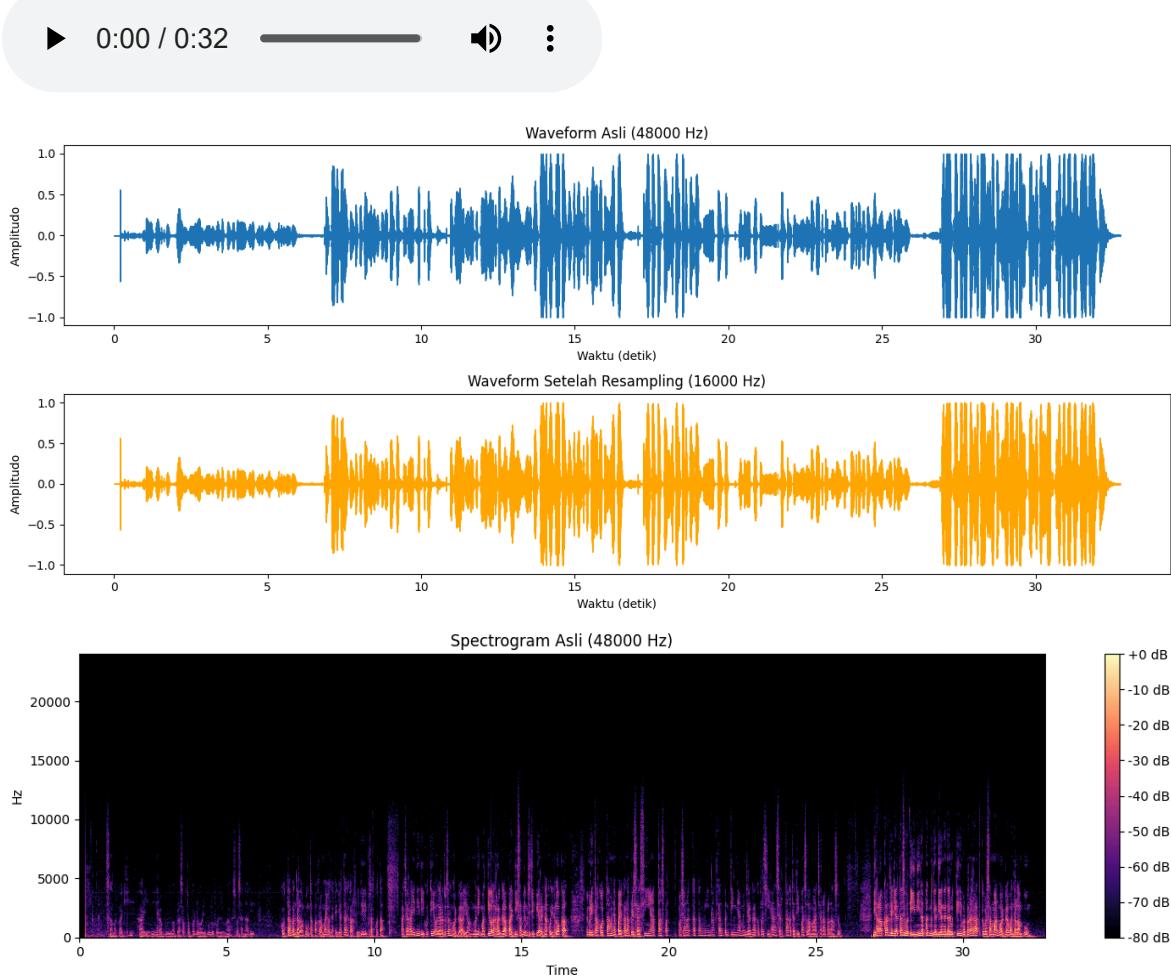
plt.tight_layout()
plt.show()

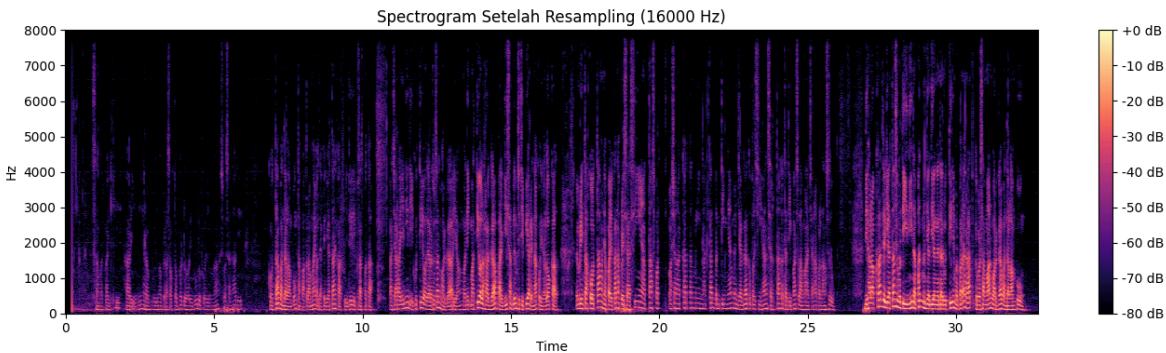
# === 4 Visualisasi Spectrogram Perbandingan ===
def plot_spec(y, sr, title):
    D = np.abs(librosa.stft(y, n_fft=2048, hop_length=512))
    DB = librosa.amplitude_to_db(D, ref=np.max)
    plt.figure(figsize=(14,4))
    librosa.display.specshow(DB, sr=sr, hop_length=512, x_axis='time', y_axis='h
    plt.colorbar(format='%.2f dB')
    plt.title(title)
    plt.tight_layout()
    plt.show()

plot_spec(y, sr, f"Spectrogram Asli ({sr} Hz)")
plot_spec(y_rs, target_sr, f"Spectrogram Setelah Resampling ({target_sr} Hz)")
```

Audio Setelah Resampling: Sample rate = 16000 Hz | Durasi = 32.76 detik

 Audio Hasil Resampling (16 kHz):





Perbandingan Waveform original dengan resampling: dari mata saya sendiri ini tidak ada perbedaan yang cukup signifikan antara 2 waveform ini, mungkin karena yang diubah cuma sample rate nya aja makanya bentuk dari amplitudo nya sendiri tidak berubah

Perbandingan Spektrogram original dengan resampling: untuk spektrogram ini baru keliatan perbedannya karena rentang nilai frekuensi setelah di resampling hanya sampai 8000hz, sedangkan original sampai 20000 yang membuat warna di spektrogram setelah resampling ke strect ke atas. untuk warna dan pola nya masih tetep sama dengan yang original.

Soal 2: Noise Reduction dengan Filtering

- Rekam suara Anda berbicara di sekitar objek yang berisik (seperti kipas angin, AC, atau mesin). – Rekaman tersebut harus berdurasi kurang lebih 10 detik.
- 1 IF4021 - Multimedia Information Processing Exercise - Audio Processing – Rekam dalam format WAV (atau konversikan ke WAV sebelum dimuat ke notebook).
- Gunakan filter equalisasi (high-pass, low-pass, dan band-pass) untuk menghilangkan noise pada rekaman tersebut.
- Lakukan eksperimen dengan berbagai nilai frekuensi cutoff (misalnya 500 Hz, 1000 Hz, 2000 Hz).
- Visualisasikan hasil dari tiap filter dan bandingkan spektrogramnya.
- Jelaskan:

 - Jenis noise yang muncul pada rekaman Anda
 - Filter mana yang paling efektif untuk mengurangi noise tersebut
 - Nilai cutoff yang memberikan hasil terbaik
 - Bagaimana kualitas suara (kejelasan ucapan) setelah proses filtering

```
In [5]: # === 💡 LANGKAH 1 – Import Library ===
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as signal
import IPython.display as ipd

# === 🎵 LANGKAH 2 – Load File Audio ===
file_path = "soal 2.wav"
y, sr = librosa.load(file_path, sr=None)
dur = len(y) / sr

print(f"File: {file_path}")
print(f"Sample rate: {sr} Hz | Durasi: {dur:.2f} detik")
```

File: soal 2.wav
Sample rate: 48000 Hz | Durasi: 8.56 detik

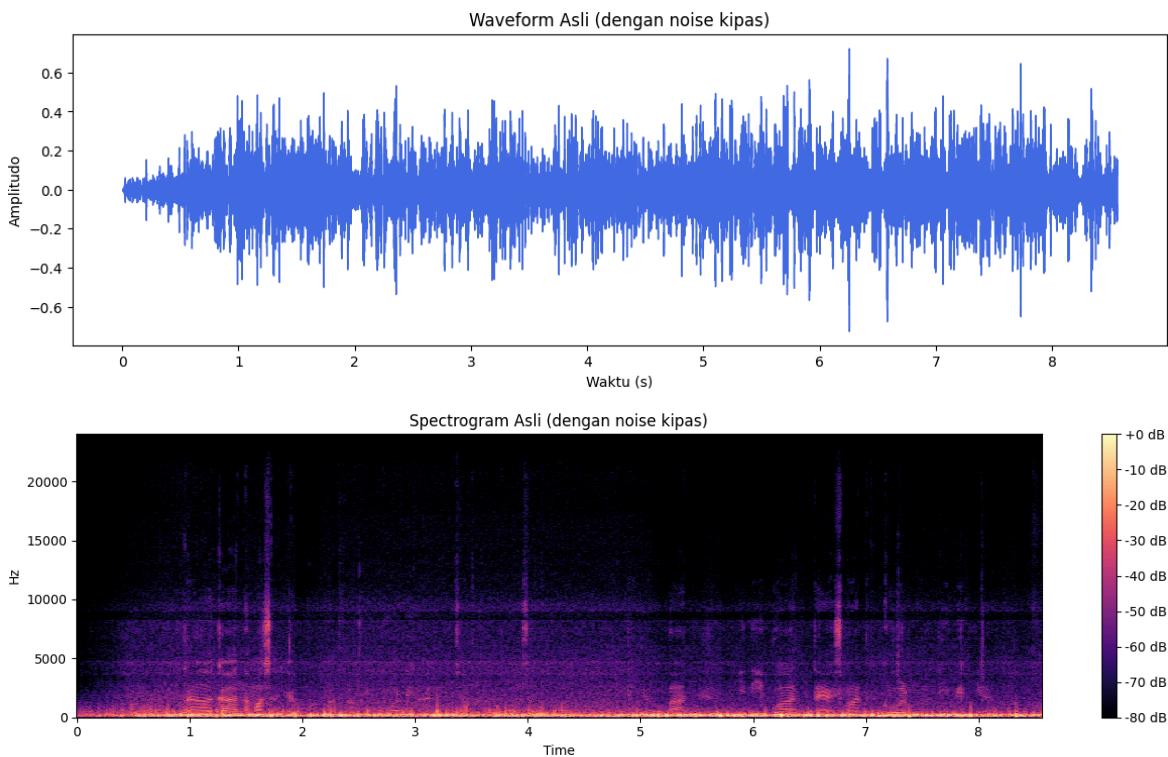
```
In [ ]: # === 📊 LANGKAH 3 – Visualisasi Awal: Waveform dan Spectrogram ===
plt.figure(figsize=(14,4))
```

```

librosa.display.waveform(y, sr=sr, color='royalblue')
plt.title("Waveform Asli (dengan noise kipas)")
plt.xlabel("Waktu (s)")
plt.ylabel("Amplitudo")
plt.show()

# Spectrogram
D = np.abs(librosa.stft(y))
DB = librosa.amplitude_to_db(D, ref=np.max)
plt.figure(figsize=(14,4))
librosa.display.specshow(DB, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
plt.colorbar(format='%.+2.0f dB')
plt.title("Spectrogram Asli (dengan noise kipas)")
plt.tight_layout()
plt.show()

```



```

In [ ]: # === 📈 LANGKAH 4 – Fungsi Filter Digital Butterworth ===
def butter_filter(data, cutoff, fs, btype, order=6):
    nyq = 0.5 * fs
    if isinstance(cutoff, (list, tuple, np.ndarray)):
        wn = [c / nyq for c in cutoff]
    else:
        wn = cutoff / nyq
    b, a = signal.butter(order, wn, btype=btype)
    filtered = signal.filtfilt(b, a, data)
    return filtered

# === 💡 LANGKAH 5 – Terapkan Berbagai Filter ===
# High-pass:
y_high = butter_filter(y, cutoff=400, fs=sr, btype='high')

# Low-pass:
y_low = butter_filter(y, cutoff=6000, fs=sr, btype='low')

# Band-pass:
y_band = butter_filter(y, cutoff=[400, 5000], fs=sr, btype='band')

```

```
# === 🎧 LANGKAH 6 – Auduo Play ===

print("\n🎙️ High-pass Filter ")
display(ipd.Audio(y_high, rate=sr))

print("🔊 Low-pass Filter ")
display(ipd.Audio(y_low, rate=sr))

print("🎤 Band-pass Filter")
display(ipd.Audio(y_band, rate=sr))
```

🎧 High-pass Filter

▶ 0:00 / 0:08 ⏸ 🔊 ⋮

🔊 Low-pass Filter

▶ 0:00 / 0:08 ⏸ 🔊 ⋮

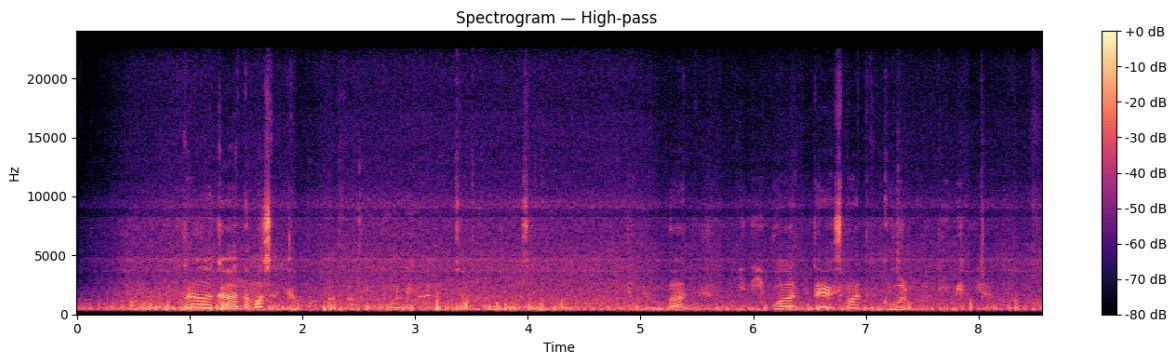
🎤 Band-pass Filter

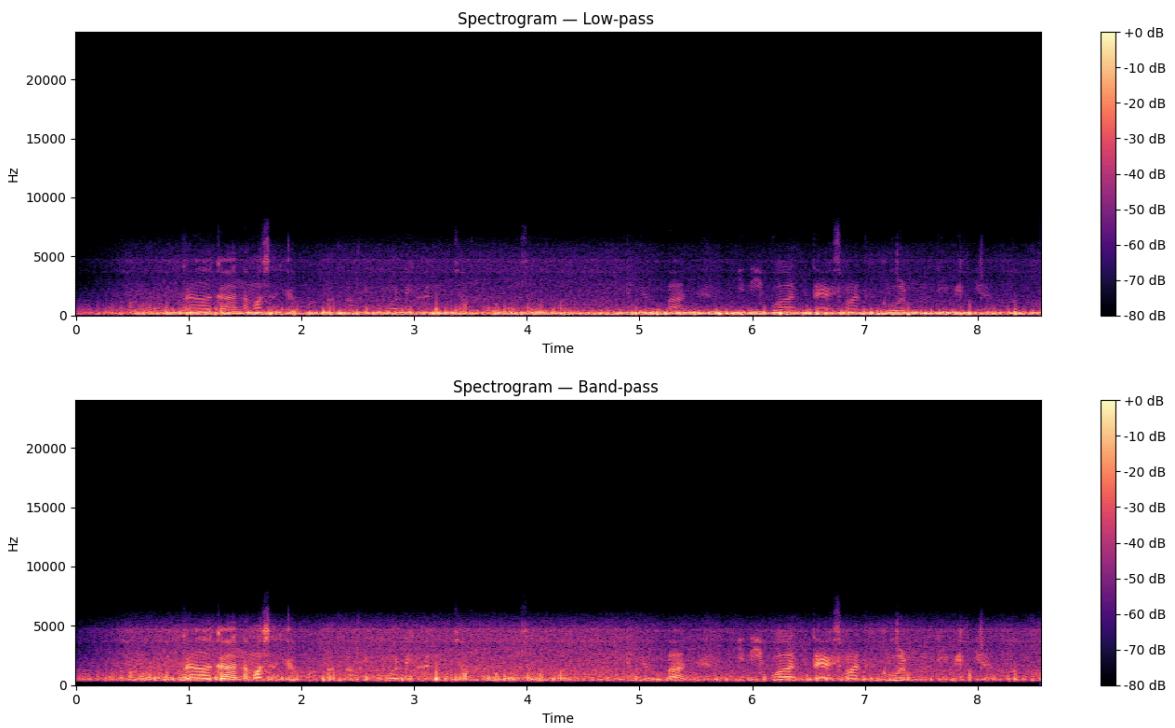
▶ 0:00 / 0:08 ⏸ 🔊 ⋮

```
In [ ]: # === 📈 LANGKAH 7 – Visualisasi Spectrogram untuk Setiap Filter ===

def plot_spec(y, sr, title):
    D = np.abs(librosa.stft(y, n_fft=2048, hop_length=512))
    DB = librosa.amplitude_to_db(D, ref=np.max)
    plt.figure(figsize=(14,4))
    librosa.display.specshow(DB, sr=sr, hop_length=512, x_axis='time', y_axis='hz')
    plt.colorbar(format='%+2.0f dB')
    plt.title(title)
    plt.tight_layout()
    plt.show()

plot_spec(y_high, sr, "Spectrogram – High-pass ")
plot_spec(y_low, sr, "Spectrogram – Low-pass ")
plot_spec(y_band, sr, "Spectrogram – Band-pass ")
```





Jenis noise yang muncul pada rekaman Anda : Noise bisa dilihat di spektrogram waktu saya tidak berbicara, ada gelombang kecil yang manandakan itu suara kipas, dan untuk jenis noise ini adalah noise broadband yang artinya bunyinya menyebar di banyak frekuensi. Filter mana yang paling efektif untuk mengurangi noise tersebut: yang paling efektif itu menggunakan band pass, karena suara manusia itu ada di rentang tengah tengah, jadi bisa menggunakan low pass buat ngilangin energi tinggi, dan high pass untuk ngilangin energi rendah. Nilai cutoff yang memberikan hasil terbaik: dari teori suara manusia itu ada di rentang 300 - 3400 Hz, jadi saya coba menggunakan high pass di 400 Hz dan low pass di 5000 Hz,

Soal 3: Pitch Shifting dan Audio Manipulation

- Lakukan pitch shifting pada rekaman suara Soal 1 untuk membuat suara terdengar seperti chipmunk (dengan mengubah pitch ke atas).
- Visualisasikan waveform dan spektrogram sebelum dan sesudah pitch shifting.
- Jelaskan proses pitch shifting yang Anda lakukan, termasuk:
 - Parameter yang digunakan
 - Perbedaan dalam representasi visual antara suara asli dan suara yang telah dimodifikasi
 - Bagaimana perubahan pitch memengaruhi kualitas dan kejelasan suara
- Gunakan dua buah pitch tinggi, misalnya pitch +7 dan pitch +12.
- Gabungkan kedua rekaman yang telah di-pitch shift ke dalam satu file audio. (Gunakan ChatGPT / AI untuk membantu Anda dalam proses ini)

```
In [6]: # === ⚡ LANGKAH 1 – Import Library ===
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt
import IPython.display as ipd

# === 🎵 LANGKAH 2 – Load Audio Asli ===
file_path = "soal 1.wav"
y, sr = librosa.load(file_path, sr=None)
dur = len(y) / sr
```

```
print(f"File: {file_path}")
print(f"Sample rate: {sr} Hz | Durasi: {dur:.2f} detik")
```

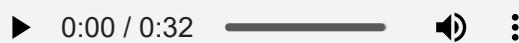
File: soal 1.wav
Sample rate: 48000 Hz | Durasi: 32.76 detik

```
In [7]: # === 🎶 LANGKAH 3 – Lakukan Pitch Shifting ===
# n_steps = jumlah semitone yang diubah (positif = naik, negatif = turun)
y_pitch_7 = librosa.effects.pitch_shift(y, sr=sr, n_steps=7)
y_pitch_12 = librosa.effects.pitch_shift(y, sr=sr, n_steps=12)

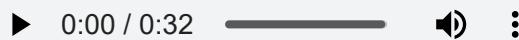
# 🔍 Dengarkan hasil pitch shifting
print("\n䴓 Pitch Shift +7 Semitone (Chipmunk ringan):")
display(ipd.Audio(y_pitch_7, rate=sr))

print("_MOUSE_ Pitch Shift +12 Semitone (Chipmunk tinggi):")
display(ipd.Audio(y_pitch_12, rate=sr))
```

䴓 Pitch Shift +7 Semitone (Chipmunk ringan):



.MOUSE_ Pitch Shift +12 Semitone (Chipmunk tinggi):



```
In [ ]: # === 🎵 LANGKAH 4 – Visualisasi Waveform Sebelum dan Sesudah ===
plt.figure(figsize=(14,8))

plt.subplot(3,1,1)
librosa.display.waveshow(y, sr=sr)
plt.title("Waveform Asli")

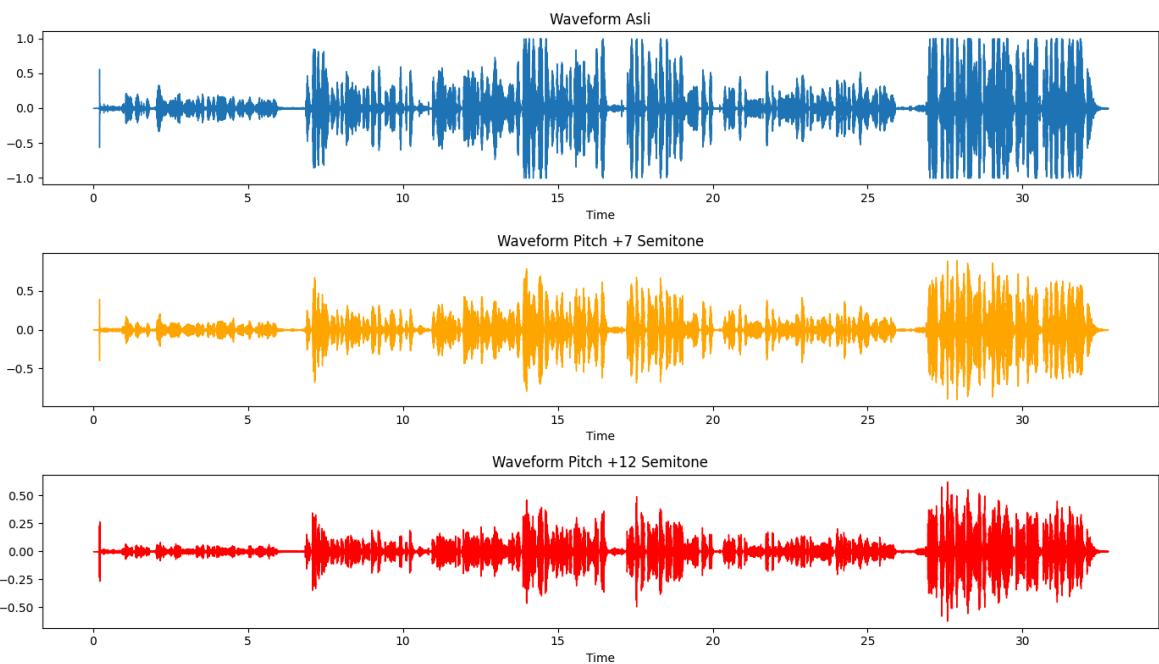
plt.subplot(3,1,2)
librosa.display.waveshow(y_pitch_7, sr=sr, color='orange')
plt.title("Waveform Pitch +7 Semitone")

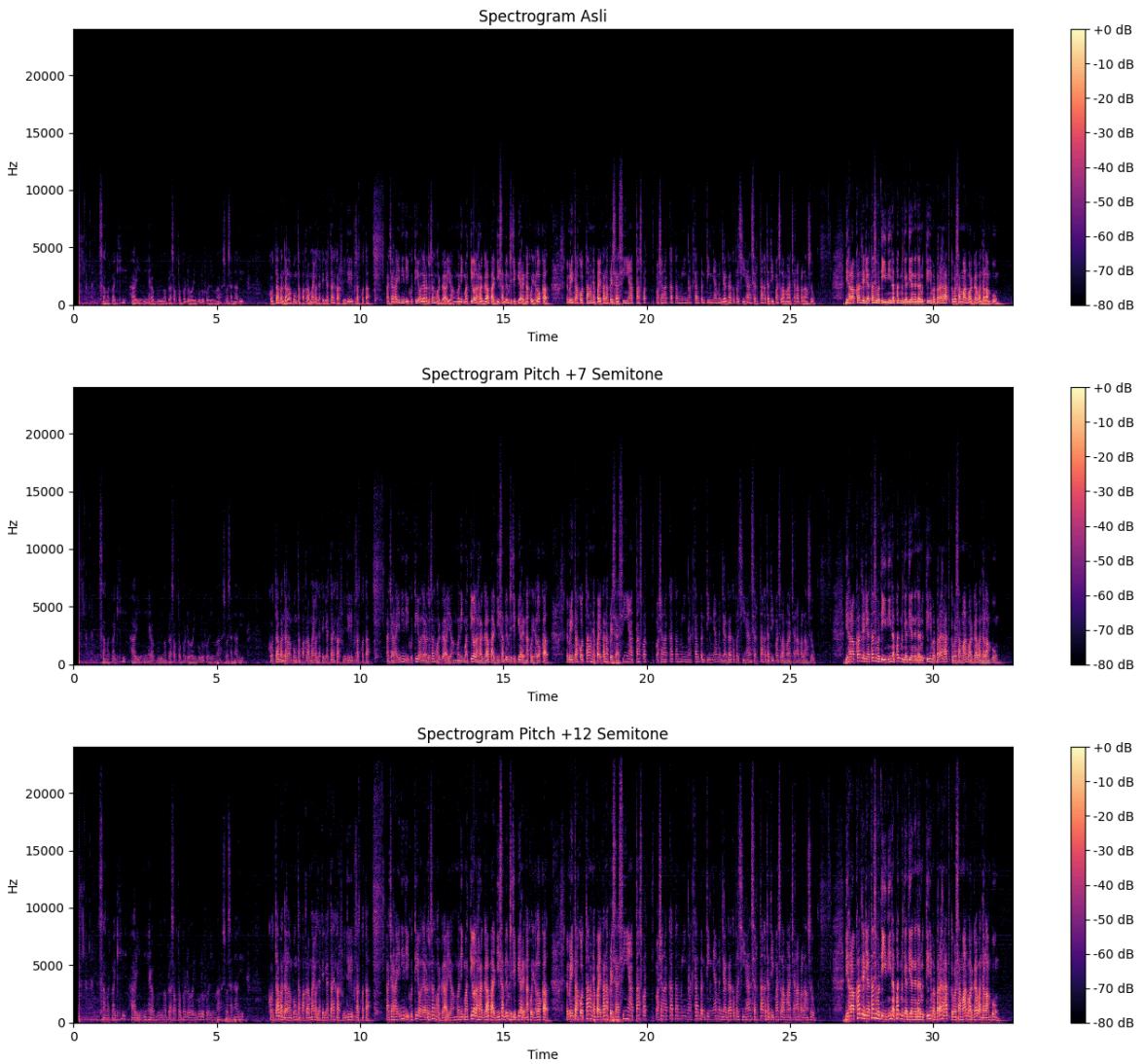
plt.subplot(3,1,3)
librosa.display.waveshow(y_pitch_12, sr=sr, color='red')
plt.title("Waveform Pitch +12 Semitone")

plt.tight_layout()
plt.show()

# === 🌈 LANGKAH 5 – Visualisasi Spectrogram ===
def plot_spec(y, sr, title):
    D = np.abs(librosa.stft(y))
    DB = librosa.amplitude_to_db(D, ref=np.max)
    plt.figure(figsize=(14,4))
    librosa.display.specshow(DB, sr=sr, x_axis='time', y_axis='hz', cmap='magma'
    plt.colorbar(format='%.2f dB')
    plt.title(title)
    plt.tight_layout()
    plt.show()

plot_spec(y, sr, "Spectrogram Asli")
plot_spec(y_pitch_7, sr, "Spectrogram Pitch +7 Semitone")
plot_spec(y_pitch_12, sr, "Spectrogram Pitch +12 Semitone")
```





Parameter yang digunakan : y : sinyal audio asli sr : sample rate rekaman n_steps : jumlah perubahan nada yang digunakan dalam satuan semitone di bagian ini yang diubah dibagian n_steps nya aja Perbedaan dalam representasi visual antara suara asli dan suara yang telah dimodifikasi : tapi waveform tidak terlalu jauh beda tapi dari hasil shifting terlihat lebih rapat gelombangnya spektogram bisa dilihat dari seluruh frekuensi naik, untuk +7 semitone frekuensi di 15000 hz ada warna di +12 semitone frekuesinya sampai ke 20000 hz

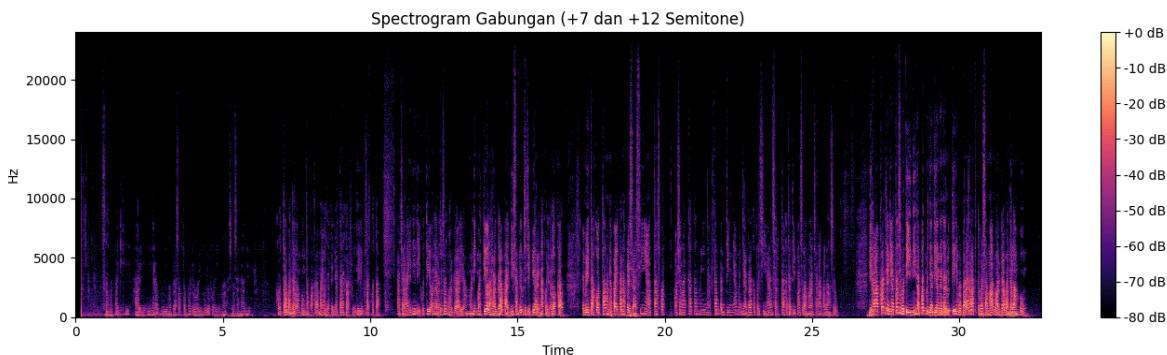
– Bagaimana perubahan pitch memengaruhi kualitas dan kejelasan suara : dari +7 semitone suara saya menjadi lebih tinggi dan kejelasan suara masih cukup jelas untuk +12 semitone suara saya sudah menjadi chipmunk dan menjadi lucu, kejelasannya sudah berkurang

```
In [ ]: # === 🎵 LANGKAH 6 – Gabungkan Dua Hasil Pitch Shift ===
# Sesuaikan panjang agar sama
min_len = min(len(y_pitch_7), len(y_pitch_12))
combined = 0.5 * (y_pitch_7[:min_len] + y_pitch_12[:min_len])

# 🎧 Dengarkan hasil kombinasi dua pitch
print("\n♪ Gabungan Pitch +7 dan +12 Semitone:")
display(ipd.Audio(combined, rate=sr))
```

♪ Gabungan Pitch +7 dan +12 Semitone:

In []: # === 🎵 LANGKAH 7 – Visualisasi Spectrogram Gabungan ===
plot_spec(combined, sr, "Spectrogram Gabungan (+7 dan +12 Semitone)")



Soal 4: Audio Processing Chain • Lakukan processing pada rekaman yang sudah di-pitch shift pada Soal 3 dengan tahapan: – Equalizer – Gain/fade – Normalization – Compression – Noise Gate – Silence trimming • Atur nilai target loudness ke -16 LUFS. • Visualisasikan waveform dan spektrogram sebelum dan sesudah proses normalisasi. • Jelaskan: – Perubahan dinamika suara yang terjadi – Perbedaan antara normalisasi peak dan normalisasi LUFS – Bagaimana kualitas suara berubah setelah proses normalisasi dan loudness optimization – Kelebihan dan kekurangan dari pengoptimalan loudness dalam konteks rekaman suara

In [9]: # === 🎵 LANGKAH 1 – Import Library ===
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt
import IPython.display as ipd
import soundfile as sf
import pyloudnorm as pyln
import scipy.signal as signal

=== 🎤 LANGKAH 2 – Load Audio ===
y = y_pitch_7
print("🎧 Menggunakan audio hasil Pitch +7 Semitone dari Soal 3")

🎧 Menggunakan audio hasil Pitch +7 Semitone dari Soal 3

In [11]: # === 📊 Fungsi Visualisasi ===
def plot_wave_spec(y, sr, title):
 plt.figure(figsize=(14,5))
 plt.subplot(2,1,1)
 librosa.display.waveshow(y, sr=sr, color='royalblue')
 plt.title(f"Waveform – {title}")
 plt.xlabel("Waktu (s)"); plt.ylabel("Amplitudo")

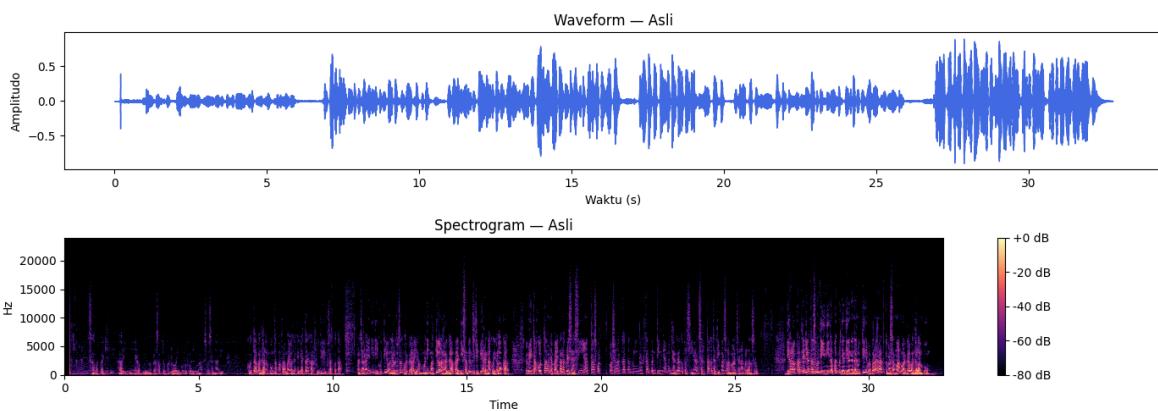
 plt.subplot(2,1,2)
 D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
 librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
 plt.colorbar(format='%+2.0f dB')
 plt.title(f"Spectrogram – {title}")

```

    plt.tight_layout()
    plt.show()

plot_wave_spec(y, sr, "Asli")

```



```

In [13]: # === ⚡ LANGKAH 3 – Equalizer ===
def simple_eq(y, sr):
    b_low, a_low = signal.butter(4, 400/(0.5*sr), btype='high')
    b_high, a_high = signal.butter(4, 6000/(0.5*sr), btype='low')
    y_eq = signal.filtfilt(b_low, a_low, y)
    y_eq = signal.filtfilt(b_high, a_high, y_eq)
    return y_eq

y_eq = simple_eq(y, sr)

```

```

In [14]: # === ⚡ LANGKAH 4 – Gain / Fade ===
y_gain = y_eq * 6.0

```

```

In [ ]: print(np.max(np.abs(y_eq)))      # amplitudo asli
         print(np.max(np.abs(y_gain)))    # amplitudo setelah gain

```

0.8019969097220746
4.811981458332448

```

In [16]: # === ⚡ LANGKAH 5 – Normalisasi Peak ===

# Cek nilai peak sebelum normalisasi
peak_before = np.max(np.abs(y_gain))
print(f"Peak sebelum normalisasi: {20*np.log10(peak_before):.2f} dBFS")

# Normalisasi Peak
y_norm_peak = y_gain / np.max(np.abs(y_gain))

# Cek nilai peak sesudah normalisasi
peak_after = np.max(np.abs(y_norm_peak))
print(f"Peak setelah normalisasi: {20*np.log10(peak_after):.2f} dBFS")

print("\n🔊 Setelah Peak Normalization:")
display(ipd.Audio(y_norm_peak, rate=sr))

```

Peak sebelum normalisasi: 13.65 dBFS
Peak setelah normalisasi: 0.00 dBFS

🔊 Setelah Peak Normalization:

In []: # === 🎧 LANGKAH 6 – Normalisasi LUFS (-16 LUFS) ===

```
meter = pyln.Meter(sr) # Loudness meter
loudness_original = meter.integrated_loudness(y_norm_peak)
print(f'Loudness sebelum normalisasi LUFS: {loudness_original:.2f} LUFS')

# Target loudness -16 LUFS
target_loudness = -16.0
loudness_difference = target_loudness - loudness_original
y_norm_lufs = y_norm_peak * (10** (loudness_difference/20))

loudness_after = meter.integrated_loudness(y_norm_lufs)
print(f'Loudness setelah normalisasi LUFS: {loudness_after:.2f} LUFS')

print("🔊 Setelah Normalisasi -16 LUFS:")
display(ipd.Audio(y_norm_lufs, rate=sr))
```

Loudness sebelum normalisasi LUFS: -19.47 LUFS

Loudness setelah normalisasi LUFS: -16.00 LUFS

🔊 Setelah Normalisasi -16 LUFS:

In []: # === 📐 LANGKAH 7 – Compression ===

```
def compressor(audio, threshold=-20, ratio=4, makeup_gain=1.2):
    compressed = []
    for sample in audio:
        if abs(sample) > 10** (threshold/20):
            sample = np.sign(sample) * (10** (threshold/20) + (abs(sample)-10** (threshold/20))/ratio)
        compressed.append(sample)
    compressed = np.array(compressed) * makeup_gain
    return compressed

y_comp = compressor(y_norm_lufs)
print("👤 Setelah Compression:")
display(ipd.Audio(y_comp, rate=sr))
```

👤 Setelah Compression:

In []: # === 🟤 LANGKAH 8 – Noise Gate ===

```
def noise_gate(y, threshold=0.01):
    return np.where(np.abs(y) < threshold, 0, y)

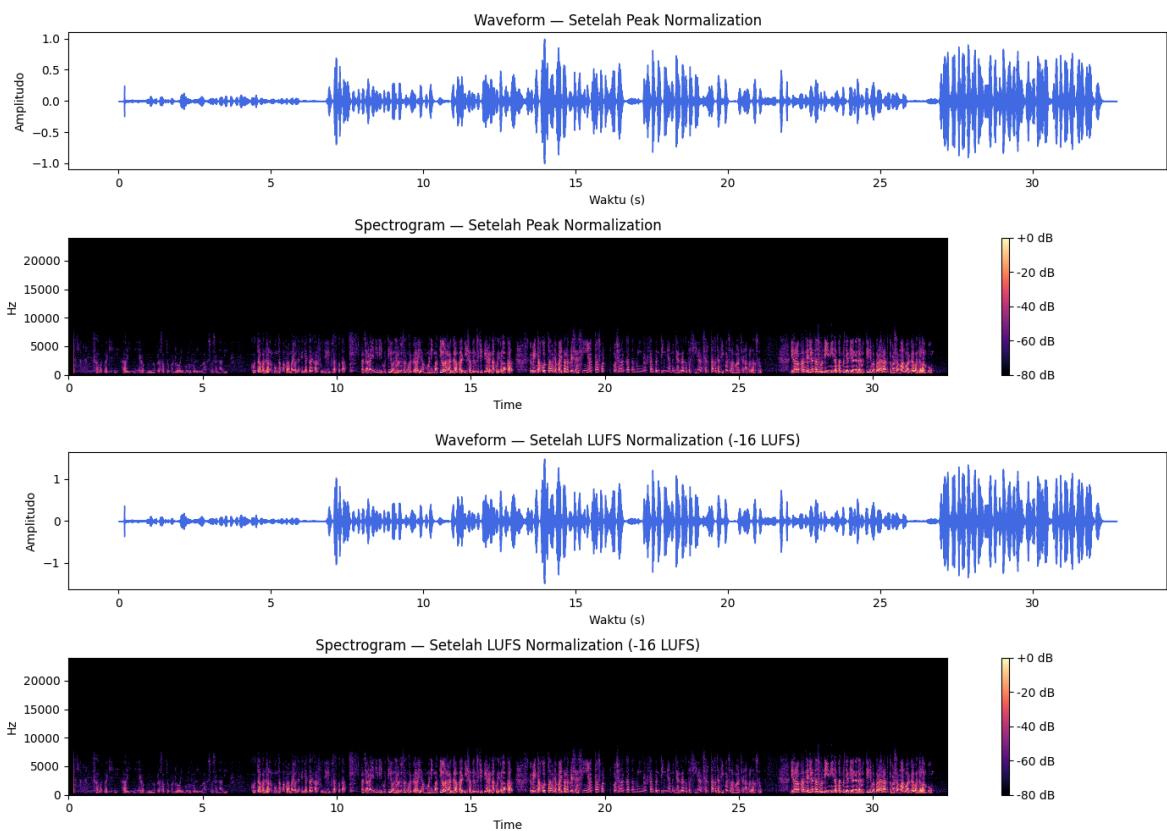
y_gate = noise_gate(y_comp)
print("🟤 Setelah Noise Gate (hilangkan noise pelan):")
display(ipd.Audio(y_gate, rate=sr))
```

🟤 Setelah Noise Gate (hilangkan noise pelan):

```
In [ ]: # === 🚫 LANGKAH 9 – Silence Trimming ===
y_trim, idx = librosa.effects.trim(y_gate, top_db=50)
print(f"🚫 Setelah Silence Trimming: durasi jadi {len(y_trim)/sr:.2f} detik")
display(ipd.Audio(y_trim, rate=sr))
```

🚫 Setelah Silence Trimming: durasi jadi 32.15 detik

```
In [ ]: # === 🎨 LANGKAH 10 – Visualisasi Sebelum & Sesudah Normalisasi ===
plot_wave_spec(y_norm_peak, sr, "Setelah Peak Normalization")
plot_wave_spec(y_norm_lufs, sr, "Setelah LUFS Normalization (-16 LUFS)")
```



Perubahan dinamika suara yang terjadi : karena tidak bisa dilihat dari waveform dan spektogram nya, saya analisis bersadarkan numerik aja sebelum di peak puncak suara masuk jauh dari atas maksimal nya di 13.65 dBFS peningkatan peak ini membuat suara rendah dan tinggi dinaikkan semuanya untuk lufs ada peningkatan dari -19.47 menjadi -16 LUFS jadi suara keseluruhan menjadi terdengar seimbang dan nyaman di telinga

Perbedaan antara normalisasi peak dan normalisasi LUFS : peak menyesuaikan puncak tertinggi audio di -dBFS, sedangkan LUFS menyesuaikan rata rata loudness dari persensi manusia dari efek pendegaran manusia peak membuat suara rendah dan tinggi menjadi naik dan turun semua yang membuat suara rendah itu menjadi tidak terdengar, sedangkan LUFS membuat suara menjadi seimbang suara rendah dan tinggi nya yang membuat suara itu menjadi lebih nyaman di dengar

Bagaimana kualitas suara berubah setelah proses normalisasi dan loudness optimization : peak membuat suara rendah dan tinggi menjadi naik dan turun semua, makanya yang terdengar lebih dominan itu suara yang tinggi untuk LUFS suara menjadi lebih rata dan seimbang antara suara rendah dan tinggi dan membuat audio menjadi lebih nyaman

Kelebihan dan kekurangan dari pengoptimalan loudness dalam konteks rekaman suara : LUFS membuat suara itu lebih konsisten di industri, profesional, dan nyaman didengar, kekurangannya jika dilakukan berlebihan akan membuat lagu itu kehilangan karakter suara, dan flat.

Soal 5: Music Analysis dan Remix • Pilih 2 buah (potongan) lagu yang memiliki vokal (penyanyi) dan berdurasi sekitar 1 menit: – Lagu 1: Nuansa sedih, lambat – Lagu 2: Nuansa ceria, cepat • Konversikan ke format WAV sebelum dimuat ke notebook. • Lakukan deteksi tempo (BPM) dan estimasi kunci (key) dari masing-masing lagu dan berikan analisis singkat. • Lakukan remix terhadap kedua lagu: – Time Stretch: Samakan tempo kedua lagu – Pitch Shift: Samakan kunci (key) kedua lagu – Crossfading: Gabungkan kedua lagu dengan efek crossfading – Filter Tambahan: Tambahkan filter kreatif sesuai keinginan (opsional) • Jelaskan proses dan parameter yang digunakan. • Tampilkan waveform dan spektrogram sesudah remix. • Jelaskan hasil remix yang telah dilakukan.

In [17]:

```
import librosa
import numpy as np
from IPython.display import Audio, display

# === 1. Load dua file audio ===
lagu1_path = 'lagu_1.wav'    # ubah ke nama file kamu
lagu2_path = 'lagu_2.wav'    # ubah ke nama file kamu

# sr=None biar sample rate asli tetap dipakai
y1, sr1 = librosa.load(lagu1_path, sr=None)
y2, sr2 = librosa.load(lagu2_path, sr=None)

# === 2. Deteksi Tempo dan Key untuk Lagu 1 ===
tempo1, _ = librosa.beat.beat_track(y=y1, sr=sr1)
tempo1 = float(np.atleast_1d(tempo1)[0])

chroma1 = librosa.feature.chroma_cqt(y=y1, sr=sr1)
key_index1 = np.mean(chroma1, axis=1).argmax()
pitch_classes = [ 'C', 'C#', 'D', 'D#', 'E', 'F',
                  'F#', 'G', 'G#', 'A', 'A#', 'B' ]

print("🎵 Lagu 1 (Sedih):")
print(f"  Tempo: {tempo1:.2f} BPM")
print(f"  Key: {pitch_classes[key_index1]} major/minor (estimasi)")

# === 3. Deteksi Tempo dan Key untuk Lagu 2 ===
tempo2, _ = librosa.beat.beat_track(y=y2, sr=sr2)
tempo2 = float(np.atleast_1d(tempo2)[0]) # ubah ke float juga

chroma2 = librosa.feature.chroma_cqt(y=y2, sr=sr2)
key_index2 = np.mean(chroma2, axis=1).argmax()

print("\n🎵 Lagu 2 (Ceria):")
```

```
print(f"    Tempo: {tempo2:.2f} BPM")
print(f"    Key: {pitch_classes[key_index2]} major/minor (estimasi)")
```

♪ Lagu 1 (Sedih):

Tempo: 95.70 BPM

Key: D major/minor (estimasi)

♪ Lagu 2 (Ceria):

Tempo: 156.61 BPM

Key: D major/minor (estimasi)

```
In [24]: # --- 2. Hitung rata-rata tempo ---
target_bpm = (tempo1 + tempo2) / 2
print(f"    Tempo target rata-rata: {target_bpm:.2f} BPM")

# --- 3. Hitung rasio time-stretch masing-masing Lagu ---
rate1 = target_bpm / tempo1 # rasio untuk Lagu 1 (lebih cepat)
rate2 = target_bpm / tempo2 # rasio untuk Lagu 2 (lebih Lambat)

print(f'Lagu 1 (sedih) rate stretch: {rate1:.3f}')
print(f'Lagu 2 (ceria) rate stretch: {rate2:.3f}')

y1_stretch = librosa.effects.time_stretch(y1, rate=rate1)
y2_stretch = librosa.effects.time_stretch(y2, rate=rate2)

if sr1 != sr2:
    y2_stretch = librosa.resample(y2_stretch, orig_sr=sr2, target_sr=sr1)
    sr2 = sr1
```

↑ Tempo target rata-rata: 126.15 BPM

Lagu 1 (sedih) rate stretch: 1.318

Lagu 2 (ceria) rate stretch: 0.806

```
In [25]: start1, end1 = 30, 78
start2, end2 = 75, 125

# Konversi ke sample index
cut1 = y1_stretch[int(start1 * sr1):int(end1 * sr1)]
cut2 = y2_stretch[int(start2 * sr2):int(end2 * sr2)]

print(f'Lagu 1: {len(cut1)/sr1:.2f} detik | Lagu 2: {len(cut2)/sr2:.2f} detik')

# === 2. Tentukan durasi crossfade (detik) ===
crossfade_dur = 5
crossfade_samples = int(sr1 * crossfade_dur)

# Pastikan durasi cukup
if len(cut1) > crossfade_samples and len(cut2) > crossfade_samples:

    # Fade-out lagu 1
    fade_out = np.linspace(1, 0, crossfade_samples)
    cut1[-crossfade_samples:] *= fade_out

    # Fade-in lagu 2
    fade_in = np.linspace(0, 1, crossfade_samples)
    cut2[:crossfade_samples] *= fade_in

    # === 3. Gabungkan dengan crossfade ===
    cross = cut1[-crossfade_samples:] + cut2[:crossfade_samples]
    remix = np.concatenate((cut1[: -crossfade_samples], cross, cut2[crossfade_samples:]))
```

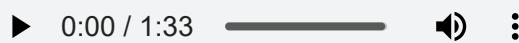
```
print(f"✅ Remix final durasi: {len(remix)/sr1:.2f} detik")
display(Audio(remix, rate=sr1))

else:
    print("❌ Durasi lagu terlalu pendek untuk crossfade segitu.")
    # === 5. Visualisasi Waveform ===
plt.figure(figsize=(12, 4))
librosa.display.waveshow(remix, sr=sr1, alpha=0.8)
plt.title("⌚ Waveform Hasil Remix (Setelah Crossfade)")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.grid(True)
plt.show()

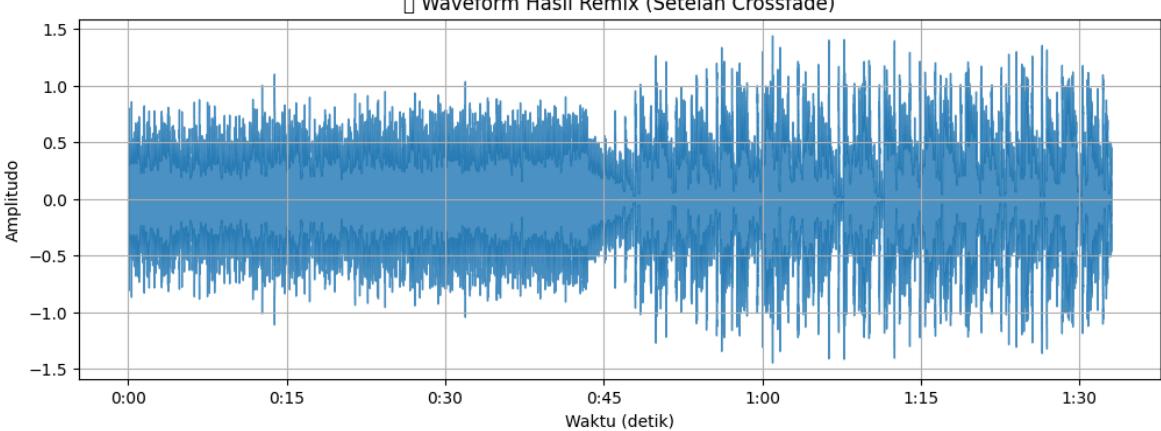
# === 6. Visualisasi Spektrogram ===
D = np.abs(librosa.stft(remix))
DB = librosa.amplitude_to_db(D, ref=np.max)

plt.figure(figsize=(12, 5))
librosa.display.specshow(DB, sr=sr1, x_axis='time', y_axis='hz', cmap='magma')
plt.colorbar(format='%.2f dB')
plt.title("🌈 Spektrogram Hasil Remix (Setelah Crossfade)")
plt.xlabel("Waktu (detik)")
plt.ylabel("Frekuensi (Hz)")
plt.show()
```

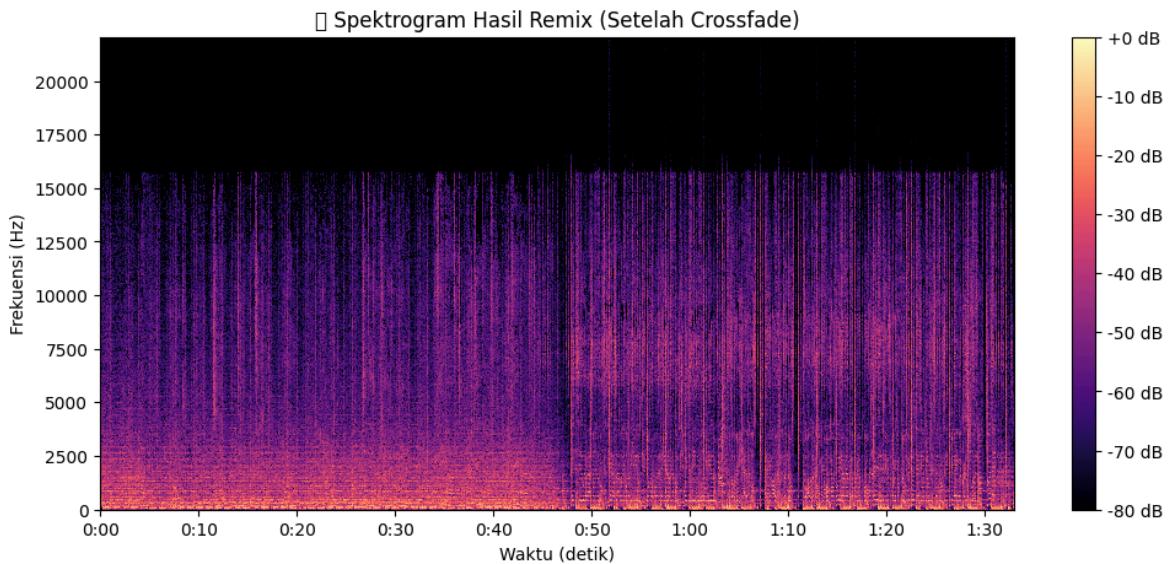
Lagu 1: 48.00 detik | Lagu 2: 50.00 detik
✓ Remix final durasi: 93.00 detik



```
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 127898 (\N{LEVEL SLIDER}) missing from font(s) DejaVu Sans.  
    fig.canvas.print_figure(bytes_io, **kw)
```



```
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 127752 (\N{RAINBOW}) missing from font(s) DejaVu Sans.  
    fig.canvas.print_figure(bytes_io, **kw)
```



Jelaskan proses dan parameter yang digunakan : proses nya:

- pertama menentukan tempo masing2 lagu dan nge rata rata in nya: target_bpm = (tempo1 + tempo2) / 2 Tempo lagu 1 (sedih) = 95.70 BPM Tempo lagu 2 (ceria) = 156.61 BPM Tempo target rata-rata = 126.15 BPM
- menyesuaikan tempo lagu masing2 lagu : y1_stretch = librosa.effects.time_stretch(y1, rate=rate1) y2_stretch = librosa.effects.time_stretch(y2, rate=rate2) Lagu 1 (sedih) dipercepat sedikit agar mendekati 126 BPM. Lagu 2 (ceria) diperlambat sedikit supaya tidak terlalu cepat.
- ketiga memotong lagu yang akan saya gunakan. cut1 = y1_stretch[int(start1 * sr1):int(end1 * sr1)] cut2 = y2_stretch[int(start2 * sr2):int(end2 * sr2)] Lagu 1 diambil dari detik ke 30–78 Lagu 2 diambil dari detik ke 75–125
- terakhir mengatur transisi antar lagu (crossfade) dengan kode crossfade_dur = 5 fade_out = np.linspace(1, 0, crossfade_samples) fade_in = np.linspace(0, 1, crossfade_samples)

parameter yang dilakukan : tempo lagu 1 untuk melihat tempo lagu 1 : 95.70 BPM tempo lagu 2 untuk melihat tempo lagu 2 : 156.61 BPM tempo target untuk hasil rata rata tempo lagu : 126.15 BPM crossfade durasi untuk durasi transisi antar lagu : 5 detik rate strect lagu 1 untuk rasio percepatan lagu 1 : 1.318 rate strect lagu 2 untuk rasio percepatan lagu 2 : 0.805

Tampilkan waveform dan spektrogram sesudah remix : dilihat dari waveform nya dari detik 0 sampai 50 detik amplitudo nya agak kecil yang menandakan lagu sedih, di detik 45 sampai 50 detik ada amplitudo yang lebih lembut yang menandakan transisi, detik 50 sampai akhir ada amplitudo yang lebih besar yang menandakan itu lagu ceria

dari spektogram terlihat dari detik 0 sampai 50 ada warna cerah di frekuensi menengah (2000 Hz), sedangkan di detik 50 sampai akhir ada warna cerah di frekuensi tinggi (7500 - 10000 Hz) terlihat juga tanda crossfade dibagian warna yang menyatu di tengah tanpa jeda frekuensi

Jelaskan hasil remix yang telah dilakukan: setelah di remix tempo nya disamain yang hasilnya lagu sedih menjadi agak cepat karena tempo nya lebih cepat setelah di rata

ratain, sedangkan di lagu ceria yang awalnya cepat temponya menjadi lambat.

link referensi : chatgpt : <https://chatgpt.com/share/68f25a7e-f04c-8013-bfbb-264972d45e6b>