🎵 Hands-On Audio Processing

Nama: Arkan Hariz Chandrawinata Liem
NIM: 122140038
Link Repositori: GitHub - Hands-on (Multimedia)

Library

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import librosa
import soundfile as sf
import scipy.signal
from IPython.display import Audio, HTML, display
import os

print("Library versions:")
print(f"NumPy: {np.__version__}")
print(f"Matplotlib: {plt.matplotlib.__version__}")
print(f"Librosa: {librosa.__version__}")
print(f"SciPy: {scipy.__version__}")
```

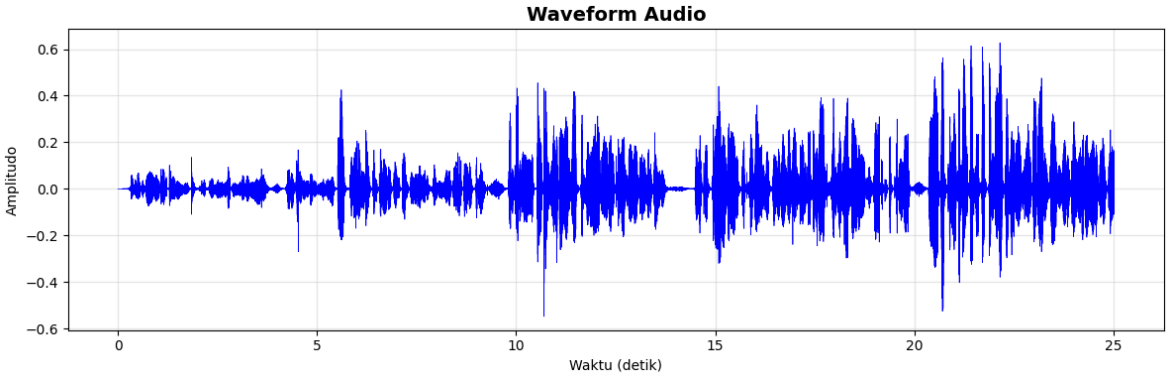
```
Library versions:
NumPy: 2.2.6
Matplotlib: 3.10.6
Librosa: 0.11.0
SciPy: 1.15.3
```

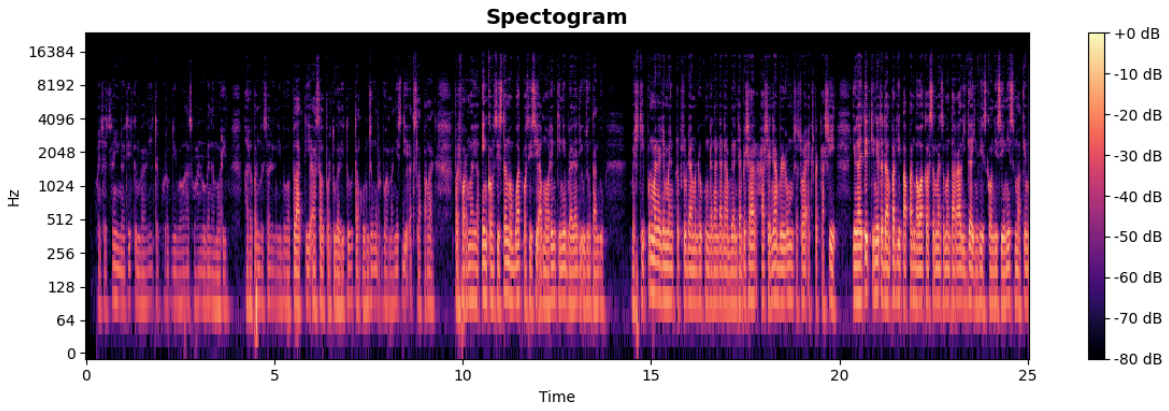
📝 Soal 1: Rekaman dan Analisis Suara Multi-Level

In [2]:
```
PATH_AUDIO_1 = os.path.join(os.getcwd(), 'data_ho', 'soal_1.wav')

if os.path.exists(PATH_AUDIO_1):
    y, sr = librosa.load(PATH_AUDIO_1, sr=None)
    source_info = f"Berhasil load audio: {PATH_AUDIO_1}"

print("🎵 AUDIO LOADED")
print("=" * 40)
print(source_info)
print(f"📊 Shape: {y.shape}")
print(f"🎵 Sample rate: {sr:,} Hz")
print(f"⏱️ Durasi: {len(y)/sr:.2f} detik")
```

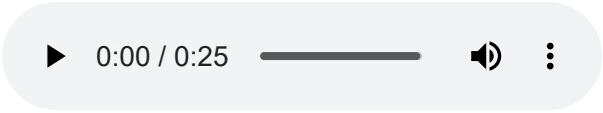
```
🎵 AUDIO LOADED
========================================
Berhasil load audio: d:\ALIEM1\Semester_7\MULTIMEDIA\PERTEMUAN_5\ho\data_ho\soal_
1.wav
📊 Shape: (1200960,)
🎵 Sample rate: 48,000 Hz
⏱️ Durasi: 25.02 detik
```

In [3]:
```python
# Waveform
fig, ax = plt.subplots(1, 1, figsize=(12, 4))
t = np.linspace(0, len(y)/sr, len(y))
ax.plot(t, y, color='blue', linewidth=0.5)
ax.set_title('Waveform Audio', fontsize=14, fontweight='bold')
ax.set_xlabel('Waktu (detik)')
ax.set_ylabel('Amplitudo')
ax.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

# Spectrogram
D = np.abs(librosa.stft(y))**2
S = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)

plt.figure(figsize=(12, 4))
librosa.display.specshow(S, sr=sr, x_axis='time', y_axis='log', cmap='magma')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectogram', fontsize=14, fontweight='bold')
plt.tight_layout()
plt.show()

# Audio player
print("🔊 Audio Player:")
display(Audio(y, rate=sr))
```





🔊 Audio Player:



Penjelasan:
Pada waveform terlihat pada 5 detik pertama, amplitudo terpantau rendah yang menandakan audio yang direkam pelan atau berbisik. 5 detik selanjutnya amplitudo perlahan-lahan mulai tinggi sampai di sekitar detik 20, amplitudo sangat tinggi dikarenakan audio yang direkam teriak. Selain itu, terdapat beberapa detik tidak ada reaksi pada amplitudo, itu menandakan orang yang merekam sedang mengambil nafas.

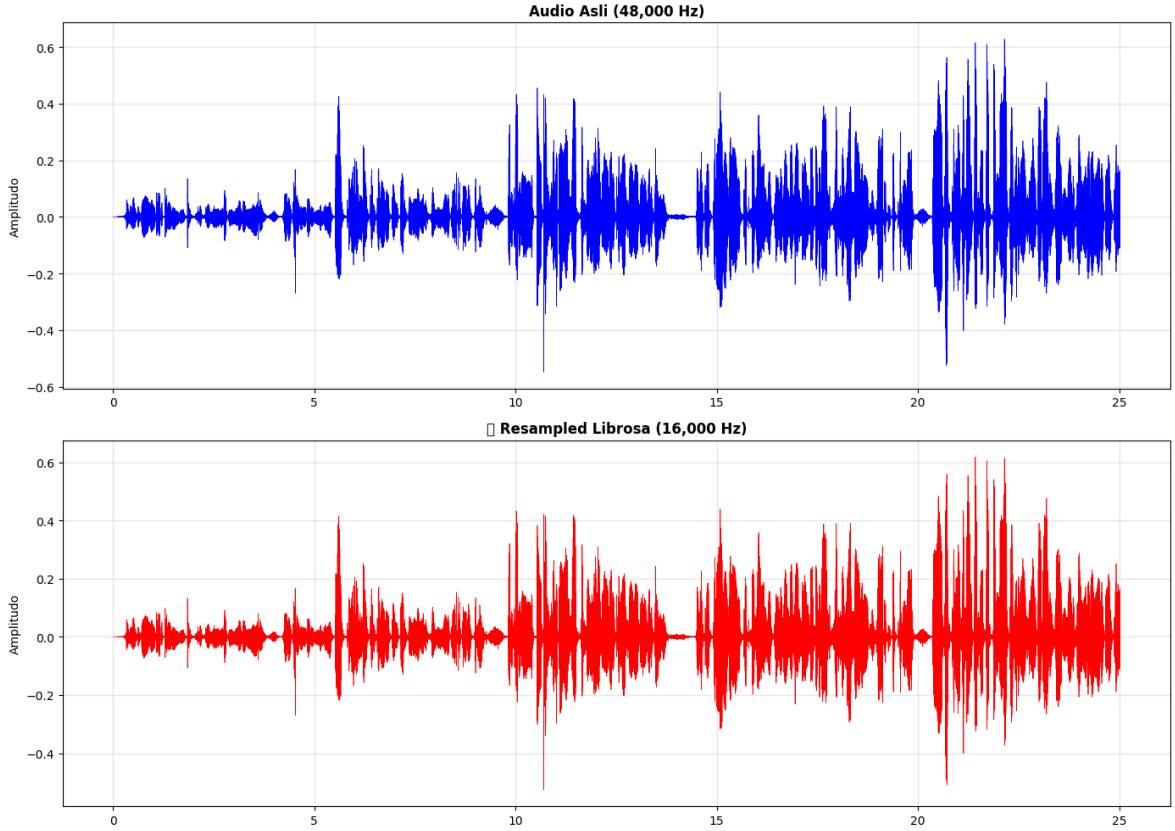
Pada spectogram terlihat energi sinyal audio tersebar cukup dominan pada frekuensi rendah hingga menengah. Pada rentang frekuensi di bawah 500 Hz tampak area dengan warna cerah yang menandakan amplitudo bermayoritas tinggi, yang menandakan suara yang dihasilkan. Secara keseluruhan, spectrogram ini menunjukkan audio dengan energi kuat di frekuensi rendah-menengah, dan relatif lebih sedikit informasi pada frekuensi tinggi.

In [ ]:
```
# Resampling ke 22 kHz
target_sr = 16000

y_resampled_librosa = librosa.resample(y, orig_sr=sr, target_sr=target_sr)

print(f"Audio asli: {sr:,} Hz, {len(y):,} samples")
print(f"Target sample rate: {target_sr:,} Hz")
print()
print("HASIL RESAMPLING:")
print(f"Librosa: {len(y_resampled_librosa):,} samples")
print(f"Durasi:  {len(y_resampled_librosa)/target_sr:.2f} detik")

fig, axes = plt.subplots(2, 1, figsize=(14, 10))

# Audio asli
t_orig = np.linspace(0, len(y)/sr, len(y))
axes[0].plot(t_orig, y, color='blue', linewidth=0.5)
axes[0].set_title(f'Audio Asli ({sr:,} Hz)', fontsize=12, fontweight='bold')
axes[0].set_ylabel('Amplitudo')
axes[0].grid(True, alpha=0.3)

# Resampled Librosa
t_resamp = np.linspace(0, len(y_resampled_librosa)/target_sr,
                       len(y_resampled_librosa))
axes[1].plot(t_resamp, y_resampled_librosa, color='red', linewidth=0.5)
axes[1].set_title(f'🔄 Resampled Librosa ({target_sr:,} Hz)', fontsize=12,
                  fontweight='bold')
axes[1].set_ylabel('Amplitudo')
axes[1].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

print("🔊 AUDIO COMPARISON:")
print("Original Audio:")
display(Audio(y, rate=sr))
```

```
print("Resampled Audio (Librosa):")
display(Audio(y_resampled_librosa, rate=target_sr))
```

```
Audio asli: 48,000 Hz, 1,200,960 samples
Target sample rate: 16,000 Hz

HASIL RESAMPLING:
Librosa: 400,320 samples
Durasi:  25.02 detik
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_3520\3696244235.py:29: UserWarning: Glyph 128260 (\N{ANTICLOCKWISE DOWNWARDS AND UPWARDS OPEN CIRCLE ARROWS}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
d:\ALIEM1\Semester_7\MULTIMEDIA\PERTEMUAN_5\ho\.venv\lib\site-packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128260 (\N{ANTICLOCKWISE DOWNWARDS AND UPWARDS OPEN CIRCLE ARROWS}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



AUDIO COMPARISON:
Original Audio:

0:00 / 0:25

Resampled Audio (Librosa):

0:00 / 0:25

Penjelasan:

Audio asli di resampling (downsampling) dari 48000 Hz menjadi 16000 Hz. Perbandingan dari segi audio, audio terdengar sedikit mendem/tidak keras ketika setelah di resampling dibandingkan audio aslinya.

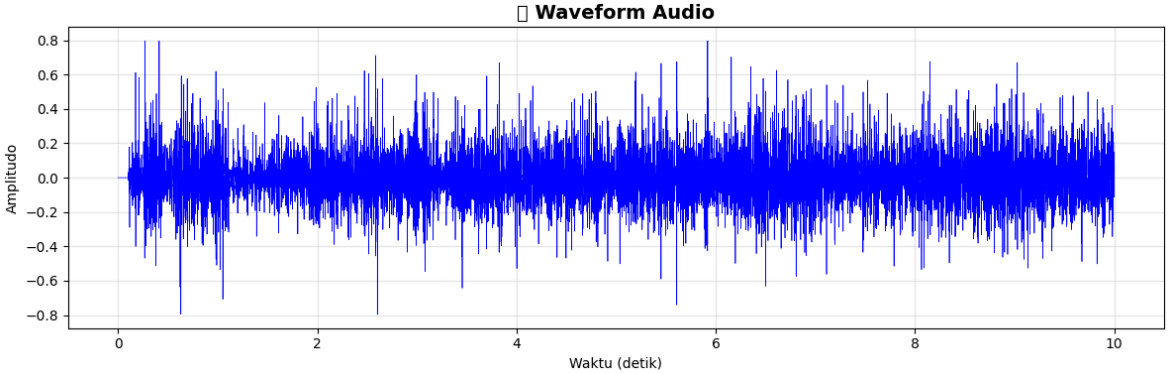
📝 Soal 2: Noise Reduction dengan Filtering

In [5]:
```
PATH_AUDIO_2 = os.path.join(os.getcwd(), 'data_ho', 'soal_2.wav')

if os.path.exists(PATH_AUDIO_2):
    y, sr = librosa.load(PATH_AUDIO_2, sr=None)
    source_info = f"Berhasil load audio: {PATH_AUDIO_2}"

print("🎵 AUDIO LOADED")
print("=" * 40)
print(source_info)
print(f"📊 Shape: {y.shape}")
print(f"🎵 Sample rate: {sr:,} Hz")
print(f"⏱️ Durasi: {len(y)/sr:.2f} detik")
```

```
🎵 AUDIO LOADED
========================================
Berhasil load audio: d:\ALIEM1\Semester_7\MULTIMEDIA\PERTEMUAN_5\ho\data_ho\soal_2.wav
📊 Shape: (480000,)
🎵 Sample rate: 48,000 Hz
⏱️ Durasi: 10.00 detik
```

In [6]:
```
fig, ax = plt.subplots(1, 1, figsize=(12, 4))
t = np.linspace(0, len(y)/sr, len(y))
ax.plot(t, y, color='blue', linewidth=0.5)
ax.set_title('🎵 Waveform Audio', fontsize=14, fontweight='bold')
ax.set_xlabel('Waktu (detik)')
ax.set_ylabel('Amplitudo')
ax.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

print("🔊 Audio Player:")
display(Audio(y, rate=sr))
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_3520\4091264272.py:8: UserWarning: Glyph 127925 (\N{MUSICAL NOTE}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
d:\ALIEM1\Semester_7\MULTIMEDIA\PERTEMUAN_5\ho\.venv\lib\site-packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 127925 (\N{MUSICAL NOTE}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



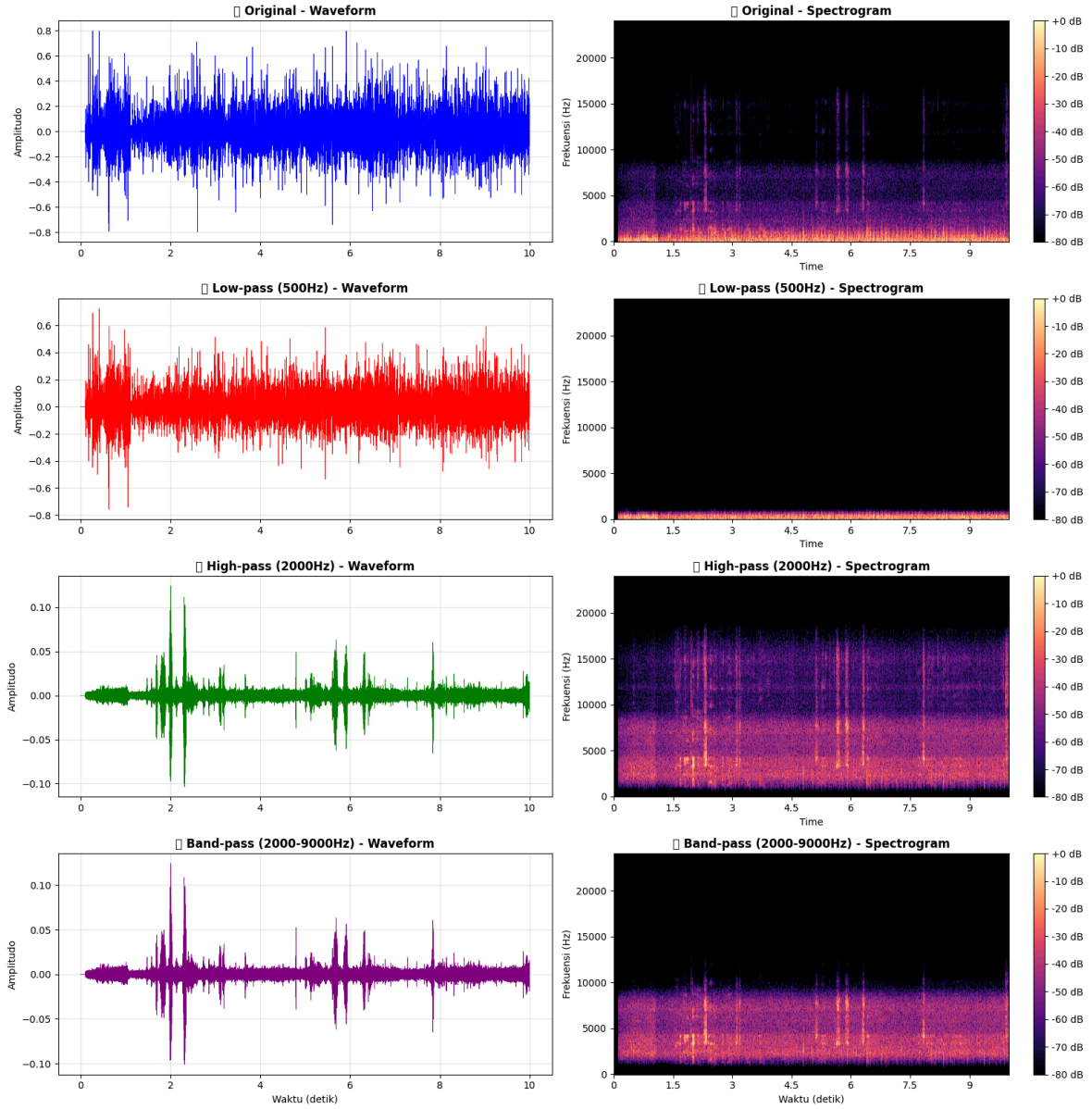
🔊 Audio Player:

In [ ]:
```
print(f"🎵 Menggunakan audio: {len(y):,} samples, {sr:,} Hz")

# Parameter filter
lowpass_cutoff = 500
highpass_cutoff = 2000
filter_order = 4

# 1. Low-pass
nyquist = sr / 2
lowpass_normalized = lowpass_cutoff / nyquist

# Membuat koefisien filter Butterworth
b_low, a_low = scipy.signal.butter(filter_order, lowpass_normalized,
                                   btype='low')

# Aplikasikan filter
audio_lowpass = scipy.signal.filtfilt(b_low, a_low, y)

print(f"✅ Low-pass filter applied: cutoff = {lowpass_cutoff} Hz")

# 2. High-pass
highpass_normalized = highpass_cutoff / nyquist
b_high, a_high = scipy.signal.butter(filter_order, highpass_normalized,
                                     btype='high')
audio_highpass = scipy.signal.filtfilt(b_high, a_high, y)

print(f"✅ High-pass filter applied: cutoff = {highpass_cutoff} Hz")

# 3. Band-pass
bandpass_low = 2000
bandpass_high = 9000
bandpass_normalized = [bandpass_low / nyquist, bandpass_high / nyquist]
b_band, a_band = scipy.signal.butter(filter_order, bandpass_normalized,
                                     btype='band')
audio_bandpass = scipy.signal.filtfilt(b_band, a_band, y)

print(f"✅ Band-pass filter applied: {bandpass_low}-{bandpass_high} Hz")

# Visualisasi comparison
fig, axes = plt.subplots(4, 2, figsize=(16, 16))

# Function untuk plot spectrogram
def plot_audio_spectrum(audio, sr, ax_wave, ax_spec, title, color):
    # Waveform
    t = np.linspace(0, len(audio)/sr, len(audio))
    ax_wave.plot(t, audio, color=color, linewidth=0.5)
    ax_wave.set_title(f'🌊 {title} - Waveform', fontweight='bold')
    ax_wave.set_ylabel('Amplitudo')
    ax_wave.grid(True, alpha=0.3)

    # Spectrogram
    D = librosa.stft(audio, n_fft=512, hop_length=128)
    mag_db = librosa.amplitude_to_db(np.abs(D), ref=np.max)
    img = librosa.display.specshow(mag_db, x_axis='time', y_axis='hz',
                                   sr=sr, hop_length=128, ax=ax_spec)
```

```
    ax_spec.set_title(f'📊 {title} - Spectrogram', fontweight='bold')
    ax_spec.set_ylabel('Frekuensi (Hz)')
    plt.colorbar(img, ax=ax_spec, format='%+2.0f dB')

# Plot semua audio
plot_audio_spectrum(y, sr, axes[0,0], axes[0,1], 'Original', 'blue')
plot_audio_spectrum(audio_lowpass, sr, axes[1,0], axes[1,1],
                    f'Low-pass ({lowpass_cutoff}Hz)', 'red')
plot_audio_spectrum(audio_highpass, sr, axes[2,0], axes[2,1],
                    f'High-pass ({highpass_cutoff}Hz)', 'green')
plot_audio_spectrum(audio_bandpass, sr, axes[3,0], axes[3,1],
                    f'Band-pass ({bandpass_low}-{bandpass_high}Hz)', 'purple')

# Set xlabel untuk bottom plots
axes[3,0].set_xlabel('Waktu (detik)')
axes[3,1].set_xlabel('Waktu (detik)')

plt.tight_layout()
plt.show()
```

```
🎵 Menggunakan audio: 480,000 samples, 48,000 Hz
✅ Low-pass filter applied: cutoff = 500 Hz
✅ High-pass filter applied: cutoff = 2000 Hz
✅ Band-pass filter applied: 2000-9000 Hz
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_3520\4048102372.py:67: UserWarning: Gl
yph 127754 (\N{WATER WAVE}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
C:\Users\USER\AppData\Local\Temp\ipykernel_3520\4048102372.py:67: UserWarning: Gl
yph 128202 (\N{BAR CHART}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
d:\ALIEM1\Semester_7\MULTIMEDIA\PERTEMUAN_5\ho\.venv\lib\site-packages\IPython\co
re\pylabtools.py:170: UserWarning: Glyph 127754 (\N{WATER WAVE}) missing from fon
t(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
d:\ALIEM1\Semester_7\MULTIMEDIA\PERTEMUAN_5\ho\.venv\lib\site-packages\IPython\co
re\pylabtools.py:170: UserWarning: Glyph 128202 (\N{BAR CHART}) missing from font
(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [8]:
```
print("🔊 AUDIO COMPARISON:")
print("Original Audio:")
display(Audio(y, rate=sr))
print(f"Low-pass Filtered ({lowpass_cutoff}Hz cutoff):")
display(Audio(audio_lowpass, rate=sr))
print(f"High-pass Filtered ({highpass_cutoff}Hz cutoff):")
display(Audio(audio_highpass, rate=sr))
print(f"Band-pass Filtered ({bandpass_low}-{bandpass_high}Hz):")
display(Audio(audio_bandpass, rate=sr))
```

🔊 AUDIO COMPARISON:
Original Audio:

0:00 / 0:10

Low-pass Filtered (500Hz cutoff):

0:00 / 0:10

High-pass Filtered (2000Hz cutoff):

0:00 / 0:10

Band-pass Filtered (2000-9000Hz):

0:00 / 0:10

Penjelasan:
– Jenis noise yang muncul pada rekaman Anda: White Noise.

– Filter mana yang paling efektif untuk mengurangi noise tersebut: Berdasarkan kasus audio saya, high-pass filter yang efektif.

– Nilai cutoff yang memberikan hasil terbaik: 2000 Hz pada High-pass filter.

– Bagaimana kualitas suara (kejelasan ucapan) setelah proses filtering: Kualitas suara percakapan cukup terdengar jelas dengan memakai High-pass filter. Suara percakapan terdengar keras dan sedikit cempreng seperti suara percakapan pada radio lama. Untuk Band-pass filter kurang lebih sama seperti High-pass filter untuk hasilnya tetapi suara percakapan dari Band-pass filter sedikit mendem/tidak keras suara walaupun masih cukup terdengar. Untuk Low-pass filter tidak efektif karena suara noise lebih terdengar jelas seperti berada di bawah air dan suara percakapan tidak terdengar.

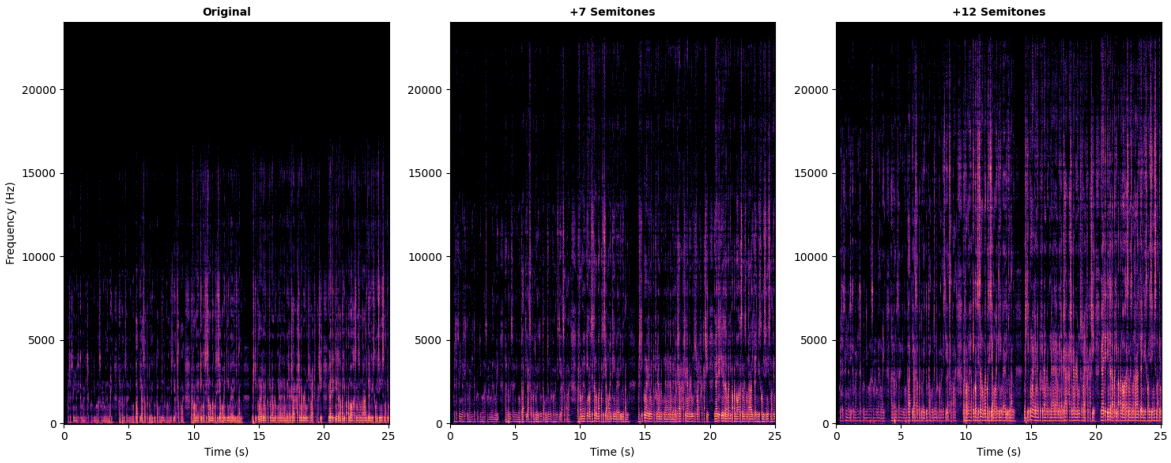
📝 Soal 3: Pitch Shifting dan Audio Manipulation

In [9]:
```
# Tambahan Library
import librosa.effects
import librosa.beat
import librosa.feature
```

In [10]:
```
PATH_AUDIO_3 = os.path.join(os.getcwd(), 'data_ho', 'soal_1.wav')

if os.path.exists(PATH_AUDIO_3):
    y, sr = librosa.load(PATH_AUDIO_3, sr=None)
    source_info = f"Berhasil load audio: {PATH_AUDIO_3}"
```

In [11]:
```
# Pitch Shifting
y_lower = librosa.effects.pitch_shift(y, sr=sr, n_steps=7)  # Menaikkan pitch 7

y_higher = librosa.effects.pitch_shift(y, sr=sr, n_steps=12) # Menaikkan pitch 1

#Visualisasi
fig, axes = plt.subplots(1, 3, figsize=(15, 6))

D_original = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
D_lower = librosa.amplitude_to_db(np.abs(librosa.stft(y_lower)), ref=np.max)
D_higher = librosa.amplitude_to_db(np.abs(librosa.stft(y_higher)), ref=np.max)
```

```
spectrograms = [
    (D_original, "Original"),
    (D_lower, "+7 Semitones"),
    (D_higher, "+12 Semitones"),
]

for i, (spec, title) in enumerate(spectrograms):
    librosa.display.specshow(spec, y_axis='hz', x_axis='time', sr=sr, ax=axes[i]
    axes[i].set_title(f'{title}', fontsize=10, fontweight='bold')
    axes[i].set_xlabel('Time (s)')
    if i == 0:
        axes[i].set_ylabel('Frequency (Hz)')
    else:
        axes[i].set_ylabel('')

plt.tight_layout()
plt.show()

print("Original:")
display(Audio(y, rate=sr))

print("+7 semitones:")
display(Audio(y_lower, rate=sr))

print("+12 semitones:")
display(Audio(y_higher, rate=sr))
```



Original:

0:00 / 0:25

+7 semitones:

0:00 / 0:25

+12 semitones:

0:00 / 0:25

Penjelasan:

Pada pitch shifting soal ini, menggunakan audio soal 1 yang dimana pitch shifting ke +7

dan +12 yang diukur dalam semitone (setengah nada). Terlihat pada ketiga spectogram, terdapat perubahan. Pada pitch +7, frekuensi asli dinaikkan/ditarik keatas. Pada pitch +12, frekuensi asli dinaikkan/ditarik keatas melebihi +7 dan pada detik awal frekuensi lebih naik dibandingkan pitch +7 dan dibeberapa titik frekuensi dinaikkan cukup tinggi.

Karena audio ini di pitch-up sehingga suara terdengar cempreng dan tinggi seperti chipmunk. Untuk kualitas terdengar lebih tipis karena suara ikut naik tetapi untuk kejelasan pada pitch +7 suara masih bisa didengar jelas dan untuk pitch +12 ada beberapa percakapan yang suara/artikulasi nya yang tidak terdengar jelas dan susah dipahami.

In [12]:
```
# Menggabungkan audio pitch +7 dan +12
min_len = min(len(y_lower), len(y_higher))
y_lower = y_lower[:min_len]
y_higher = y_higher[:min_len]

combined = (y_lower + y_higher) / 2.0

# Path output ke folder data_ho
output_path = os.path.join("data_ho", "gabungan_pitch.wav")

sf.write(output_path, combined, sr)
```

In [13]:
```
# Menyimpan suara pitch +7
output_path = os.path.join("data_ho", "pitch_+7.wav")
sf.write(output_path, y_lower, sr)
```

📝 Soal 4: Audio Processing Chain

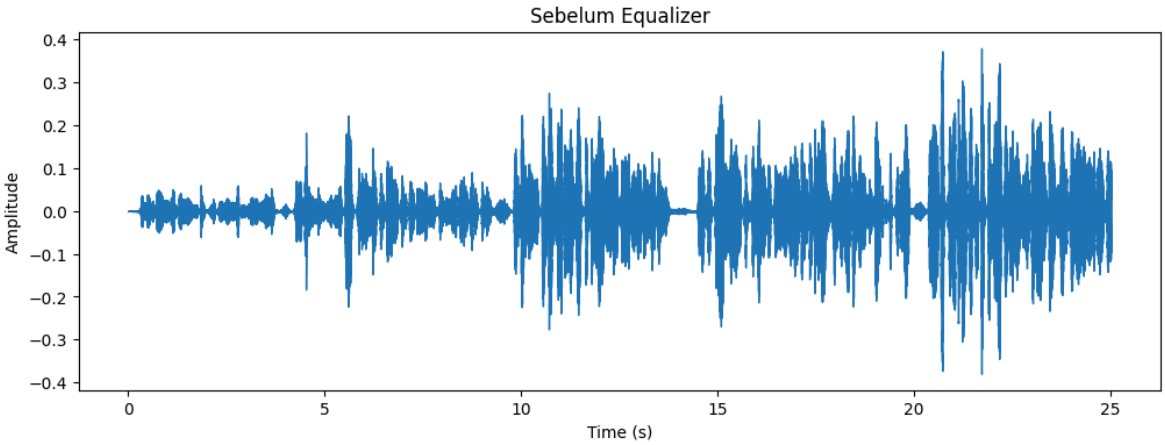
In [14]:
```
PATH_AUDIO_4 = os.path.join(os.getcwd(), 'data_ho', 'pitch_+7.wav')
```

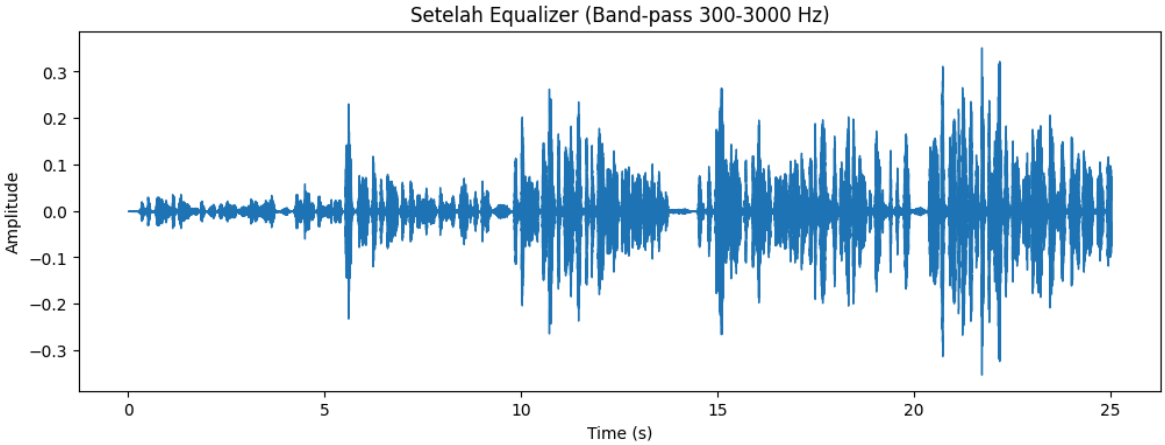
In [15]:
```
import pyloudnorm as pyln
meter = pyln.Meter(sr)  # default K-weighting
lufs = meter.integrated_loudness(y_lower)

print(f"LUFS (Integrated Loudness): {lufs:.2f} LUFS")
```

```
LUFS (Integrated Loudness): -28.36 LUFS
```

In [ ]:
```
# Equalizer (Band-pass)
bandpass_low = 300
bandpass_high = 3000
bandpass_normalized = [bandpass_low / nyquist, bandpass_high / nyquist]
b_band, a_band = scipy.signal.butter(filter_order, bandpass_normalized,
                                     btype='band')
audio_equalizer = scipy.signal.filtfilt(b_band, a_band, y_lower)

plt.figure(figsize=(12, 4))
librosa.display.waveshow(y_lower, sr=sr)
plt.title('Sebelum Equalizer')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.show()
```

```
# Setelah Equalizer
plt.figure(figsize=(12, 4))
librosa.display.waveshow(audio_equalizer, sr=sr)
plt.title('Setelah Equalizer (Band-pass 300-3000 Hz)')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.show()


print("+7 semitones")
display(Audio(PATH_AUDIO_4, rate=sr))
print("Pitch +7 semitones setelah equalizer")
display(Audio(audio_equalizer, rate=sr))
```





+7 semitones

0:00 / 0:25

Pitch +7 semitones setelah equalizer

0:00 / 0:25

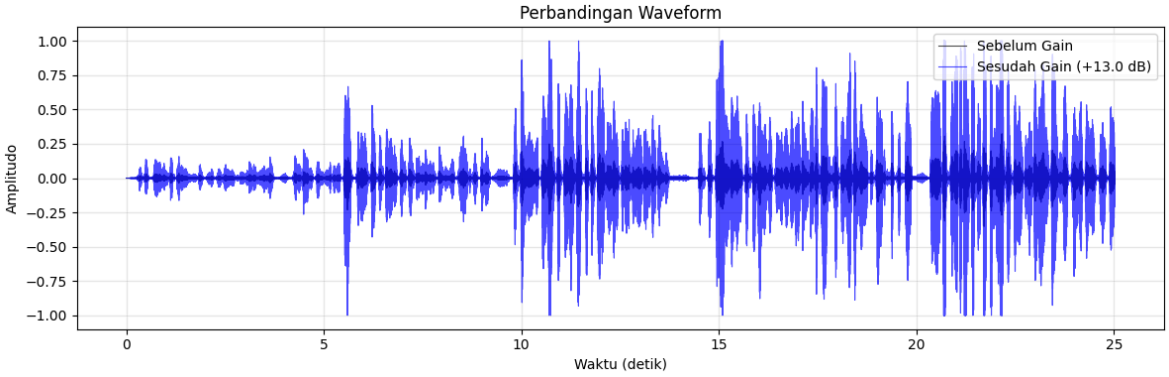
In [17]:
```
# Gain Adjustment (+13 dB) Target -16 LUFS
def peak_dbfs(sig):
    p = np.max(np.abs(sig))
    return p, 20*np.log10(p + 1e-12)

def rms_dbfs(sig):
    r = np.sqrt(np.mean(sig**2))
```

```
    return r, 20*np.log10(r + 1e-12)

def lufs_value(sig, sr):
    meter = pyln.Meter(sr)
    return meter.integrated_loudness(sig)

peak_val_before, peak_before_db = peak_dbfs(audio_equalizer)
rms_val_before, rms_before_db   = rms_dbfs(audio_equalizer)
lufs_before = lufs_value(audio_equalizer, sr)

def adjust_gain(audio_data, gain_db=13.0):
    gain_linear = 10 ** (gain_db / 20.0)
    adjusted_audio = audio_data * gain_linear
    adjusted_audio = np.clip(adjusted_audio, -1.0, 1.0) # Potong untuk mencegah
    return adjusted_audio

gain_db = 13.0
y_gain = adjust_gain(audio_equalizer, gain_db)

peak_val_after, peak_after_db = peak_dbfs(y_gain)
rms_val_after, rms_after_db   = rms_dbfs(y_gain)
lufs_after = lufs_value(y_gain, sr)
```

In [ ]:

```
print(f"Gain diterapkan: {gain_db:+.1f} dB \n")
print("Loudness (Sebelum → Sesudah):")
print(f"  • Peak: {peak_before_db:6.2f} dBFS  → {peak_after_db:6.2f} dBFS")
print(f"  • RMS : {rms_before_db:6.2f} dBFS  → {rms_after_db:6.2f} dBFS")
print(f"  • LUFS: {lufs_before:6.2f} LUFS  → {lufs_after:6.2f} LUFS")

t_full = np.linspace(0, len(audio_equalizer)/sr, len(audio_equalizer), endpoint=

plt.figure(figsize=(12, 4))
plt.plot(t_full, audio_equalizer, label="Sebelum Gain", color="black",
         alpha=0.7, linewidth=0.8)
plt.plot(t_full, y_gain, label=f"Sesudah Gain ({gain_db:+.1f} dB)",
         color="blue", alpha=0.7, linewidth=0.8)

plt.title("Perbandingan Waveform")
plt.xlabel("Waktu (detik)")
plt.ylabel("Amplitudo")
plt.grid(True, alpha=0.3)
plt.legend(loc="upper right")
plt.tight_layout()
plt.show()

print("Sebelum Gain:")
display(Audio(audio_equalizer, rate=sr))
print("Sesudah Gain (+13 dB):")
display(Audio(y_gain, rate=sr))
```

```
Gain diterapkan: +13.0 dB

Loudness (Sebelum → Sesudah):
  • Peak:  -9.06 dBFS  →   0.00 dBFS
  • RMS : -30.48 dBFS  → -17.51 dBFS
  • LUFS: -29.33 LUFS  → -16.36 LUFS
```



Sebelum Gain:

0:00 / 0:25

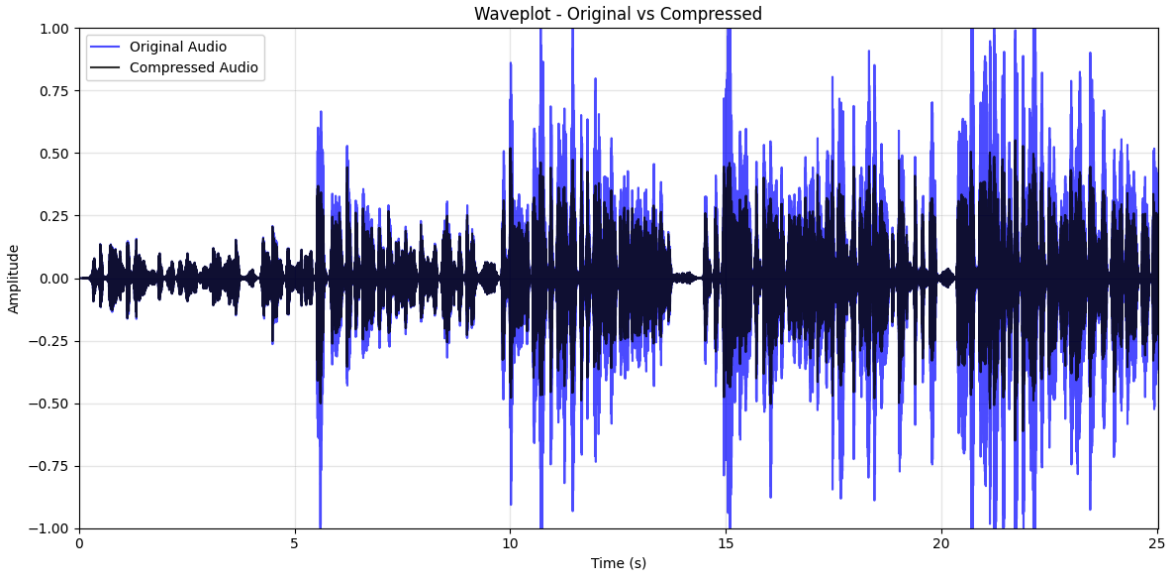
Sesudah Gain (+13 dB):

0:00 / 0:25

In [ ]:
```python
# Compression (Threshold -20 dB, Ratio 4:1, Attack 5ms, Release 50ms)
def compress_audio(audio_data, sr, threshold_db=-20.0, ratio=4.0, attack_ms=5.0,
    threshold_linear = 10 ** (threshold_db / 20.0)
    attack_samples = int((attack_ms / 1000.0) * sr)
    release_samples = int((release_ms / 1000.0) * sr)

    compressed_audio = np.zeros_like(audio_data)
    gain_reduction = 1.0
    envelope = 0.0

    for n in range(len(audio_data)):
        input_level = abs(audio_data[n])
        if input_level > threshold_linear:
            desired_gain = (threshold_linear + (input_level - threshold_linear)
        else:
            desired_gain = 1.0

        if desired_gain < gain_reduction:
            gain_reduction += (desired_gain - gain_reduction) * (1 - np.exp(-1 /
        else:
            gain_reduction += (desired_gain - gain_reduction) * (1 - np.exp(-1 /

        compressed_audio[n] = audio_data[n] * gain_reduction

    compressed_audio = np.clip(compressed_audio, -1.0, 1.0)
    return compressed_audio

compress_audio = compress_audio(y_gain, sr, threshold_db=-20.0, ratio=4.0,
                                attack_ms=5.0, release_ms=50.0)
```

In [ ]:
```python
# Visualisasi
plt.figure(figsize=(12, 6))

# Time axis for visualization
time_axis = np.linspace(0, len(y_gain) / sr, len(y_gain))

# Plot both waveforms on the same subplot
```

```
plt.plot(time_axis, y_gain, color='blue', label='Original Audio', alpha=0.7)
plt.plot(time_axis, compress_audio, color='black',
         label='Compressed Audio', alpha=0.8)

plt.title('Waveplot - Original vs Compressed')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.grid(alpha=0.3)
plt.xlim(0, len(y_gain)/sr)
plt.ylim(-1, 1)
plt.legend()
plt.tight_layout()
plt.show()

print("Gain Audio:")
display(Audio(y_gain, rate=sr))
print("Compressed Audio:")
display(Audio(compress_audio, rate=sr))
```



Gain Audio:

0:00 / 0:25

Compressed Audio:

0:00 / 0:25

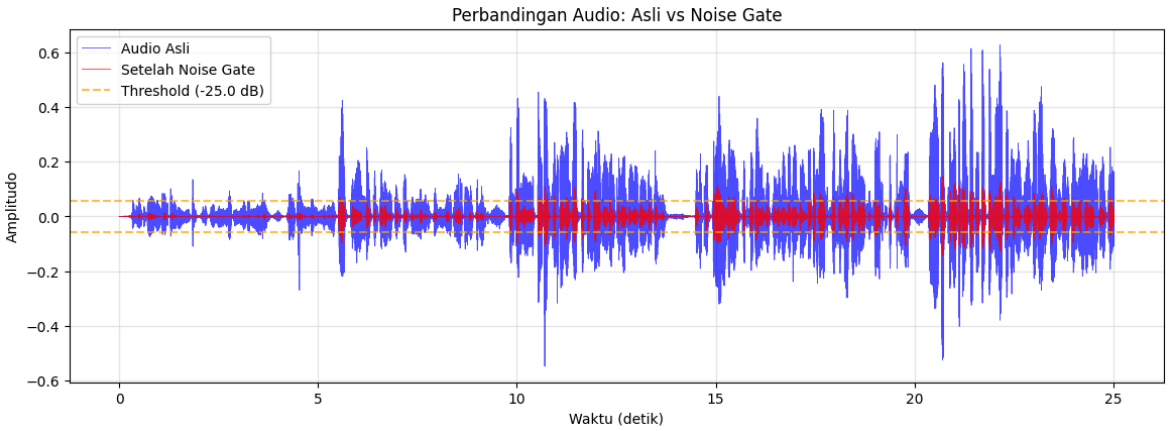
In [ ]:

```
# Noise Gate (Threshold -30 dB)
def noise_gate(audio_data, sr, threshold_db=-30.0,
               attack_ms=5.0, release_ms=50.0):
    threshold_linear = 10 ** (threshold_db / 20.0)
    attack_samples = int((attack_ms / 1000.0) * sr)
    release_samples = int((release_ms / 1000.0) * sr)

    gated_audio = np.zeros_like(audio_data)
    gate_state = 0.0

    for n in range(len(audio_data)):
        input_level = abs(audio_data[n])
        if input_level < threshold_linear:
```

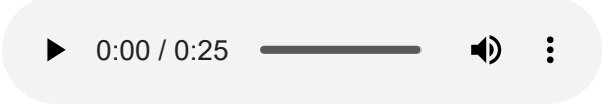
```
            desired_gate = 0.0
        else:
            desired_gate = 1.0

        if desired_gate < gate_state:
            gate_state += (desired_gate - gate_state) * (1 - np.exp(-1 / attack_
        else:
            gate_state += (desired_gate - gate_state) * (1 - np.exp(-1 / release

        gated_audio[n] = audio_data[n] * gate_state

    return gated_audio

# Terapkan noise gate
threshold_db = -25.0
gated_audio = noise_gate(compress_audio, sr, threshold_db=threshold_db,
                         attack_ms=5.0, release_ms=50.0)
```

In [ ]:

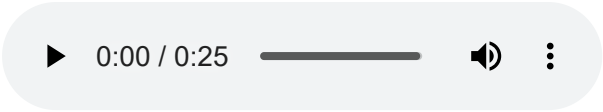
```
# Visualisasi
plt.figure(figsize=(14, 10))

plt.subplot(2, 1, 1)
plt.plot(time_axis, y, color='blue', alpha=0.7, linewidth=0.6,
         label='Audio Asli')
plt.plot(time_axis, gated_audio, color='red', alpha=0.8, linewidth=0.6, label='S
plt.axhline(y=10**(threshold_db/20.0), color='orange', linestyle='--',
            alpha=0.7, label=f'Threshold ({threshold_db} dB)')
plt.axhline(y=-10**(threshold_db/20.0), color='orange', linestyle='--',
            alpha=0.7)
plt.title('Perbandingan Audio: Asli vs Noise Gate')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.legend()
plt.grid(True, alpha=0.3)

plt.show()

print("Sebelum noise gate:")
display(Audio(compress_audio, rate=sr))
print("Sesudah noise gate:")
display(Audio(gated_audio, rate=sr))
```



Sebelum noise gate:



Sesudah noise gate:

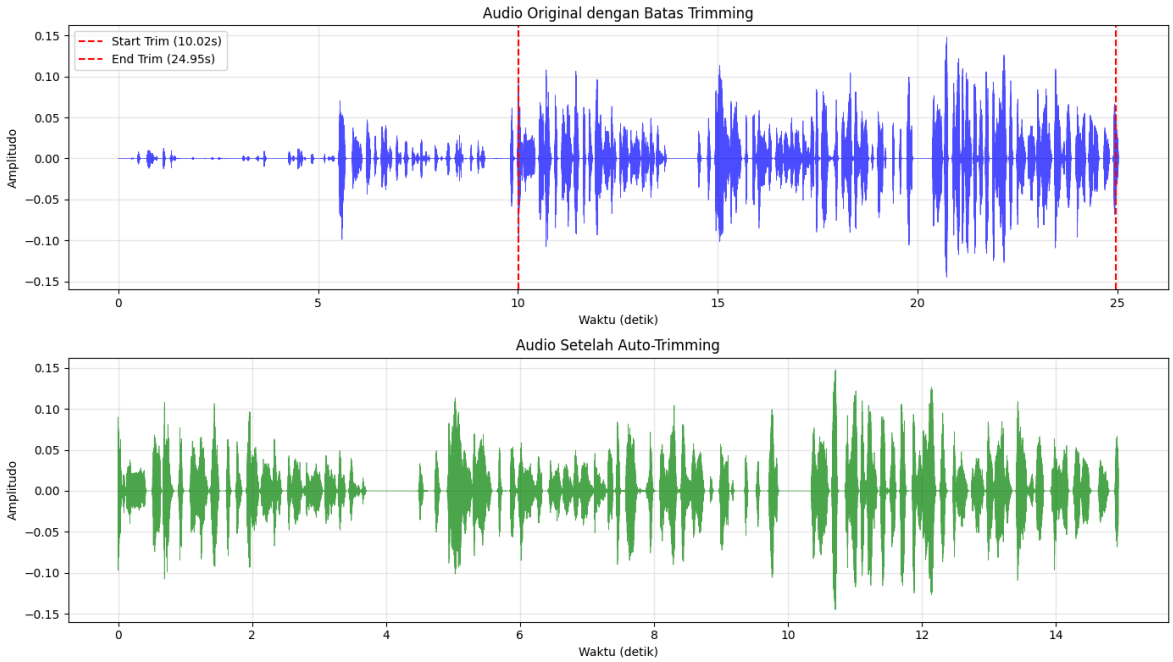


In [ ]:
```python
def trim_silence(audio_data, sr, threshold_db=-30.0, padding_ms=0):
    threshold_linear = 10 ** (threshold_db / 20.0)

    hop_length = 512
    frame_length = 1024
    rms_frames = librosa.feature.rms(y=audio_data, frame_length=frame_length,
                                     hop_length=hop_length)[0]

    # Cari frame pertama yang di atas threshold (start)
    start_frame = 0
    for i, rms_val in enumerate(rms_frames):
        if rms_val > threshold_linear:
            start_frame = i
            break

    # Cari frame terakhir yang di atas threshold (end)
    end_frame = len(rms_frames) - 1
    for i in range(len(rms_frames) - 1, -1, -1):
        if rms_frames[i] > threshold_linear:
            end_frame = i
            break

    start_sample = max(0, start_frame * hop_length)
    end_sample = min(len(audio_data), (end_frame + 1) * hop_length)

    padding_samples = int(padding_ms * sr / 1000)
    start_sample = max(0, start_sample - padding_samples)
    end_sample = min(len(audio_data), end_sample + padding_samples)

    trimmed_audio = audio_data[start_sample:end_sample]

    return trimmed_audio, start_sample, end_sample
```

In [ ]:
```python
trimmed_audio, start_idx, end_idx = trim_silence(gated_audio, sr,
                                                 threshold_db=-30.0, padding_ms=
```

In [ ]:
```python
# Plot
original_duration = len(gated_audio) / sr
trimmed_duration = len(trimmed_audio) / sr
start_time = start_idx / sr
end_time = end_idx / sr
removed_start = start_time
removed_end = original_duration - end_time

# Visualisasi perbandingan
plt.figure(figsize=(14, 8))

# Plot 1: Audio original
plt.subplot(2, 1, 1)
time_original = np.linspace(0, len(gated_audio)/sr, len(gated_audio))
plt.plot(time_original, gated_audio, color='blue', alpha=0.7, linewidth=0.5)
plt.axvline(start_time, color='red', linestyle='--',
            label=f'Start Trim ({start_time:.2f}s)')
```

```
plt.axvline(end_time, color='red', linestyle='--',
            label=f'End Trim ({end_time:.2f}s)')
plt.title('Audio Original dengan Batas Trimming')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.legend()
plt.grid(True, alpha=0.3)

plt.subplot(2, 1, 2)
time_trimmed = np.linspace(0, len(trimmed_audio)/sr, len(trimmed_audio))
plt.plot(time_trimmed, trimmed_audio, color='green', alpha=0.7, linewidth=0.5)
plt.title('Audio Setelah Auto-Trimming')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

print("Audio sebelum trimming:")
display(Audio(gated_audio, rate=sr))
print("Audio setelah trimming:")
display(Audio(trimmed_audio, rate=sr))
```



Audio sebelum trimming:

0:00 / 0:25

Audio setelah trimming:

0:00 / 0:14

Penjelasan:

– Perubahan dinamika suara yang terjadi: Setelah dilakukan pitch shifting sebesar +7 semitone, suara menjadi lebih tinggi dan tajam dibandingkan rekaman aslinya. Frekuensi meningkat sehingga karakter suara terdengar lebih ringan, namun sedikit kehilangan

kedalaman dan naturalitas. Selain itu, dinamika suara dapat berkurang karena proses pitch shifting memengaruhi amplitudo dan detail transien suara.

– Perbedaan antara normalisasi peak dan normalisasi LUFS: Normalisasi peak menyesuaikan level tertinggi dari sinyal audio agar tidak melebihi batas tertentu untuk mencegah distorsi, tanpa memperhatikan persepsi keras-lembut suara oleh telinga manusia. Sementara itu, normalisasi LUFS menyesuaikan rata-rata kenyaringan audio berdasarkan persepsi pendengaran manusia, sehingga hasilnya lebih konsisten antar file meskipun nilai puncak (peak) berbeda.

– Bagaimana kualitas suara berubah setelah proses normalisasi dan loudness optimization: Setelah proses normalisasi LUFS dan optimasi loudness, volume audio menjadi lebih stabil dan seimbang, membuat rekaman terdengar lebih jelas dan profesional. Namun, penggunaan kompresi yang berlebihan untuk mencapai target loudness dapat membuat suara kehilangan dinamika alami dan terasa lebih datar.

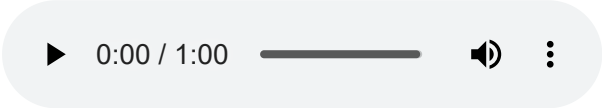
– Kelebihan dan kekurangan dari pengoptimalan loudness dalam konteks rekaman suara: Kelebihan utama pengoptimalan loudness adalah menjaga konsistensi volume antar rekaman, membuat audio lebih nyaman didengar di berbagai perangkat, dan sesuai standar platform modern. Namun, kekurangannya adalah potensi hilangnya dinamika alami, munculnya distorsi atau clipping, serta risiko membuat suara terdengar datar dan melelahkan bagi pendengar.

📝 Soal 5: Music Analysis dan Remix

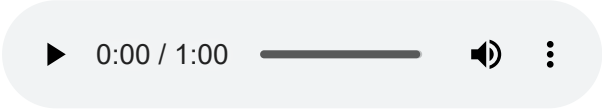
In [26]:
```
# Load
PATH_SAD_SONG = os.path.join(os.getcwd(), 'data_ho', 'lagu_1.wav')
PATH_HAPPY_SONG = os.path.join(os.getcwd(), 'data_ho', 'lagu_2.wav')

if os.path.exists(PATH_SAD_SONG) and os.path.exists(PATH_HAPPY_SONG):
    y_sad, sr_sad = librosa.load(PATH_SAD_SONG, sr=None)
    y_happy, sr_happy = librosa.load(PATH_HAPPY_SONG, sr=None)
    source_info = f"Berhasil load audio: {PATH_SAD_SONG} ({len(y_sad)} samples,
    source_info = f"Berhasil load audio: {PATH_HAPPY_SONG} ({len(y_happy)} sampl

print("🎵 AUDIO LOADED")
print("=" * 40)
print(source_info)
print("SAD SONG :(")
print(f"📊 Shape: {y_sad.shape}")
print(f"🎵 Sample rate: {sr_sad:,} Hz")
print(f"⏱️ Durasi: {len(y_sad)/sr_sad:.2f} detik")
print()
print("HAPPY SONG :)")
print(f"📊 Shape: {y_happy.shape}")
print(f"🎵 Sample rate: {sr_happy:,} Hz")
print(f"⏱️ Durasi: {len(y_happy)/sr_happy:.2f} detik")
print()
print("🔊 Audio Player - SAD SONG :(")
display(Audio(y_sad, rate=sr_sad))
print("🔊 Audio Player - HAPPY SONG :)")
display(Audio(y_happy, rate=sr_happy))
```

```
🎵 AUDIO LOADED
========================================
Berhasil load audio: d:\ALIEM1\Semester_7\MULTIMEDIA\PERTEMUAN_5\ho\data_ho\lagu_
2.wav (2646000 samples, 44100 Hz)
SAD SONG :(
📊 Shape: (2646000,)
🎵 Sample rate: 44,100 Hz
⏱️ Durasi: 60.00 detik

HAPPY SONG :)
📊 Shape: (2646000,)
🎵 Sample rate: 44,100 Hz
⏱️ Durasi: 60.00 detik
```

🔊 Audio Player - SAD SONG :(



🔊 Audio Player - HAPPY SONG :)



In [27]:
```python
# Deteksi tempo
tempo_sad, beats_sad = librosa.beat.beat_track(y=y_sad, sr=sr_sad)
tempo_happy, beats_happy = librosa.beat.beat_track(y=y_happy, sr=sr_happy)

tempo_sad = tempo_sad[0] if isinstance(tempo_sad, np.ndarray) else tempo_sad
tempo_happy = tempo_happy[0] if isinstance(tempo_happy, np.ndarray) else tempo_h

print(f"HASIL DETEKSI TEMPO:")
print()
print(f"SAD SONG :(")
print(f"BPM Terdeteksi: {tempo_sad:.1f} BPM")
print(f"Jumlah beat: {len(beats_sad)} beats")
print()
print(f"HAPPY SONG :)")
print(f"BPM Terdeteksi: {tempo_happy:.1f} BPM")
print(f"Jumlah beat: {len(beats_happy)} beats")
```

```
HASIL DETEKSI TEMPO:

SAD SONG :(
BPM Terdeteksi: 58.1 BPM
Jumlah beat: 57 beats

HAPPY SONG :)
BPM Terdeteksi: 99.4 BPM
Jumlah beat: 90 beats
```

Analisis Singkat: Berdasarkan hasil deteksi tempo, Sad Song memiliki tempo sekitar 58,1 BPM, menunjukkan ritme yang sangat lambat dan tenang, sesuai dengan karakter lagu yang cenderung melankolis dan emosional. Tempo yang lambat ini memberikan ruang bagi ekspresi dan nuansa perasaan sedih untuk lebih terasa. Sebaliknya, Happy Song memiliki tempo 99,4 BPM, hampir dua kali lebih cepat, menciptakan kesan yang lebih enerjik, ceria, dan dinamis.

In [28]:
```
# Estimasi Kunci (key)
chroma_sad = librosa.feature.chroma_stft(y=y_sad, sr=sr_sad)
chroma_mean_sad = np.mean(chroma_sad, axis=1)

chroma_happy = librosa.feature.chroma_stft(y=y_happy, sr=sr)
chroma_mean_happy = np.mean(chroma_happy, axis=1)

# Key templates (Krumhansl-Schmuckler profiles)
major_template = np.array([6.35, 2.23, 3.48, 2.33, 4.38, 4.09,
                           2.52, 5.19, 2.39, 3.66, 2.29, 2.88])
minor_template = np.array([6.33, 2.68, 3.52, 5.38, 2.60, 3.53,
                           2.54, 4.75, 3.98, 2.69, 3.34, 3.17])

major_template /= np.sum(major_template)
minor_template /= np.sum(minor_template)

# Key names
keys = ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B']

# Calculate correlations for all keys
major_scores_sad = []
minor_scores_sad = []

major_scores_happy = []
minor_scores_happy = []

for i in range(12):
    # Rotate template for each key
    major_rotated = np.roll(major_template, i)
    minor_rotated = np.roll(minor_template, i)

    # Calculate correlation
    major_corr_sad = np.corrcoef(chroma_mean_sad, major_rotated)[0, 1]
    minor_corr_sad = np.corrcoef(chroma_mean_sad, minor_rotated)[0, 1]

    major_corr_happy = np.corrcoef(chroma_mean_happy, major_rotated)[0, 1]
    minor_corr_happy = np.corrcoef(chroma_mean_happy, minor_rotated)[0, 1]

    major_scores_sad.append(major_corr_sad)
    minor_scores_sad.append(minor_corr_sad)

    major_scores_happy.append(major_corr_happy)
    minor_scores_happy.append(minor_corr_happy)

# Find best key
best_major_sad = np.argmax(major_scores_sad)
best_minor_sad = np.argmax(minor_scores_sad)

best_major_happy = np.argmax(major_scores_happy)
best_minor_happy = np.argmax(minor_scores_happy)

if major_scores_sad[best_major_sad] > minor_scores_sad[best_minor_sad]:
    detected_key_sad = f"{keys[best_major_sad]} Major"
    confidence_sad = major_scores_sad[best_major_sad]
else:
    detected_key_sad = f"{keys[best_minor_sad]} Minor"
    confidence_sad = minor_scores_sad[best_minor_sad]

if major_scores_happy[best_major_happy] > minor_scores_happy[best_minor_happy]:
```

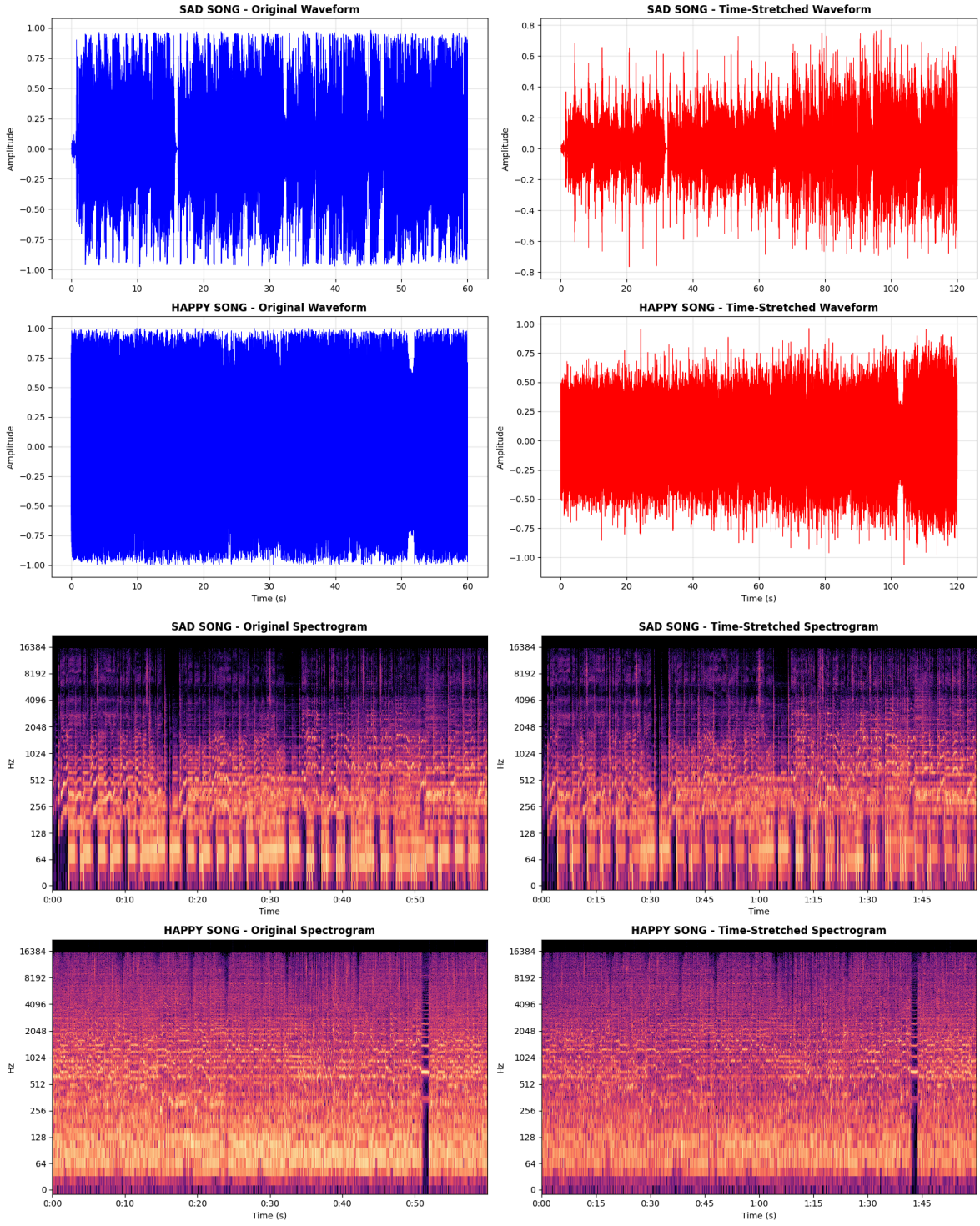
```
    detected_key_happy = f"{keys[best_major_happy]} Major"
    confidence_happy = major_scores_happy[best_major_happy]
else:
    detected_key_happy = f"{keys[best_minor_happy]} Minor"
    confidence_happy = minor_scores_happy[best_minor_happy]

print(f"DETEKSI KUNCI LAGU")
print()
print(f"SAD SONG :(")
print(f"Kunci terdeteksi: {detected_key_sad}")
print(f"Confidence: {confidence_sad:.3f}")
print()
print("HAPPY SONG :)")
print(f"Kunci terdeteksi: {detected_key_happy}")
print(f"Confidence: {confidence_happy:.3f}")
```

```
DETEKSI KUNCI LAGU

SAD SONG :(
Kunci terdeteksi: A# Major
Confidence: 0.443

HAPPY SONG :)
Kunci terdeteksi: A Minor
Confidence: 0.789
```

Analisis Singkat: Berdasarkan hasil deteksi kunci lagu, Sad Song memiliki kunci A# Major dengan tingkat kepercayaan 0.443, sedangkan Happy Song memiliki kunci A Minor dengan tingkat kepercayaan lebih tinggi, yaitu 0.789. Meskipun secara teori kunci A# Major biasanya menghasilkan nuansa cerah dan optimis, tingkat confidence yang rendah menunjukkan bahwa karakter harmoninya mungkin tidak sepenuhnya stabil atau bercampur dengan nada-nada minor, sehingga tetap menimbulkan kesan sedih. Sebaliknya, A Minor pada Happy Song justru memberikan warna yang lebih hangat dan emosional, namun tetap bisa terdengar ceria tergantung pada ritme dan instrumen pengiringnya.

In [ ]:
```
# Time Strecthing (Diperlambat 2x)
y_sad_stretch = librosa.effects.time_stretch(y_sad, rate=0.5)
y_happy_stretch = librosa.effects.time_stretch(y_happy, rate=0.5)

# Visualisasi waveform sebelum dan sesudah time stretching
fig, axes = plt.subplots(2, 2, figsize=(16, 10))

# SAD SONG
# Original
t_sad = np.linspace(0, len(y_sad)/sr_sad, len(y_sad))
axes[0, 0].plot(t_sad, y_sad, color='blue', linewidth=0.5)
axes[0, 0].set_title('SAD SONG - Original Waveform', fontweight='bold')
axes[0, 0].set_ylabel('Amplitude')
axes[0, 0].grid(True, alpha=0.3)
# Stretched
t_sad_stretch = np.linspace(0, len(y_sad_stretch)/sr_sad, len(y_sad_stretch))
axes[0, 1].plot(t_sad_stretch, y_sad_stretch, color='red', linewidth=0.5)
axes[0, 1].set_title('SAD SONG - Time-Stretched Waveform', fontweight='bold')
axes[0, 1].set_ylabel('Amplitude')
axes[0, 1].grid(True, alpha=0.3)
```

```
# HAPPY SONG
# Original
t_happy = np.linspace(0, len(y_happy)/sr_happy, len(y_happy))
axes[1, 0].plot(t_happy, y_happy, color='blue', linewidth=0.5)
axes[1, 0].set_title('HAPPY SONG - Original Waveform', fontweight='bold')
axes[1, 0].set_ylabel('Amplitude')
axes[1, 0].grid(True, alpha=0.3)
# Stretched
t_happy_stretch = np.linspace(0, len(y_happy_stretch)/sr_happy,
                              len(y_happy_stretch))
axes[1, 1].plot(t_happy_stretch, y_happy_stretch, color='red', linewidth=0.5)
axes[1, 1].set_title('HAPPY SONG - Time-Stretched Waveform', fontweight='bold')
axes[1, 1].set_ylabel('Amplitude')
axes[1, 1].grid(True, alpha=0.3)

# Set xlabel for bottom plots
axes[1, 0].set_xlabel('Time (s)')
axes[1, 1].set_xlabel('Time (s)')
plt.tight_layout()
plt.show()

# Visualisasi spectrogram sebelum dan sesudah time stretching
fig, axes = plt.subplots(2, 2, figsize=(16, 10))

# SAD SONG
# Original
D_sad = librosa.amplitude_to_db(np.abs(librosa.stft(y_sad)), ref=np.max)
librosa.display.specshow(D_sad, sr=sr_sad, x_axis='time', y_axis='log',
                         ax=axes[0, 0], cmap='magma')
axes[0, 0].set_title('SAD SONG - Original Spectrogram', fontweight='bold')
# Stretched
D_sad_stretch = librosa.amplitude_to_db(np.abs(librosa.stft(y_sad_stretch)),
                                        ref=np.max)
librosa.display.specshow(D_sad_stretch, sr=sr_sad, x_axis='time', y_axis='log',
                         ax=axes[0, 1], cmap='magma')
axes[0, 1].set_title('SAD SONG - Time-Stretched Spectrogram', fontweight='bold')

# HAPPY SONG
# Original
D_happy = librosa.amplitude_to_db(np.abs(librosa.stft(y_happy)), ref=np.max)
librosa.display.specshow(D_happy, sr=sr_happy, x_axis='time', y_axis='log',
                         ax=axes[1, 0], cmap='magma')
axes[1, 0].set_title('HAPPY SONG - Original Spectrogram', fontweight='bold')
# Stretched
D_happy_stretch = librosa.amplitude_to_db(np.abs(librosa.stft(y_happy_stretch)),
librosa.display.specshow(D_happy_stretch, sr=sr_happy, x_axis='time',
                         y_axis='log', ax=axes[1, 1], cmap='magma')
axes[1, 1].set_title('HAPPY SONG - Time-Stretched Spectrogram', fontweight='bold

# Set xlabel for bottom plots
axes[1, 0].set_xlabel('Time (s)')
axes[1, 1].set_xlabel('Time (s)')
plt.tight_layout()
plt.show()
```



In [30]:
```
print("SAD SONG - Original :(")
display(Audio(y_sad, rate=sr_sad))
print("SAD SONG - Time-Stretched :(")
display(Audio(y_sad_stretch, rate=sr_sad))
print("HAPPY SONG - Original :)")
display(Audio(y_happy, rate=sr_happy))
print("HAPPY SONG - Time-Stretched :)")
display(Audio(y_happy_stretch, rate=sr_happy))
```

```
SAD SONG - Original :(
```

0:00 / 1:00

```
SAD SONG - Time-Stretched :(
```

0:00 / 2:00

HAPPY SONG - Original :)

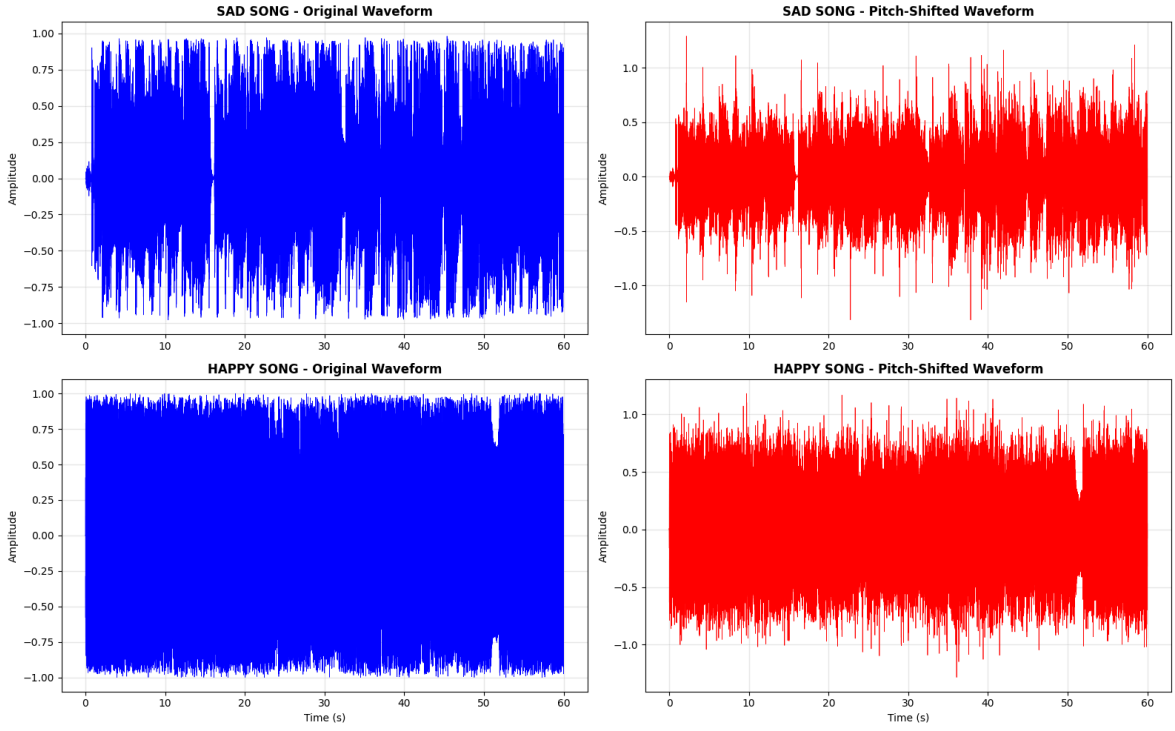
0:00 / 1:00

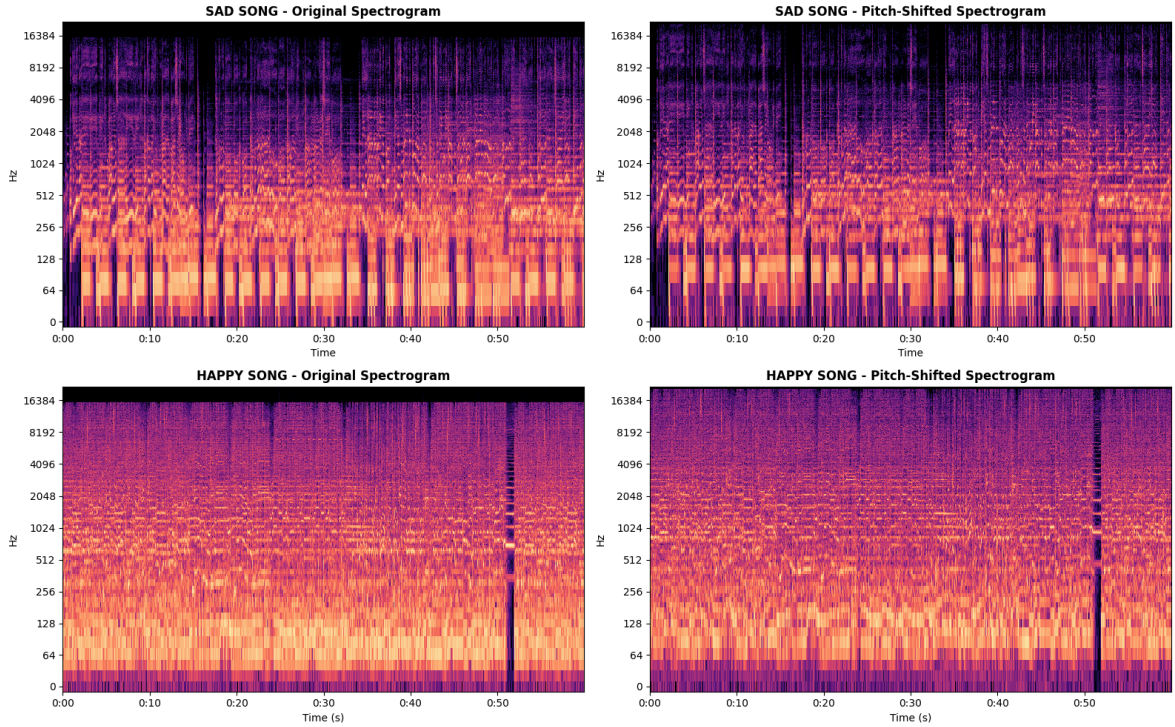
HAPPY SONG - Time-Stretched :)

0:00 / 2:00

In [ ]:
```python
# Pitch Shifting (Naik 5 semitone)
y_sad_pitch = librosa.effects.pitch_shift(y_sad, sr=sr_sad, n_steps=5)
y_happy_pitch = librosa.effects.pitch_shift(y_happy, sr=sr_happy, n_steps=5)

# Visualisasi waveform sebelum dan sesudah pitch shifting
fig, axes = plt.subplots(2, 2, figsize=(16, 10))

# SAD SONG
# Original
t_sad = np.linspace(0, len(y_sad)/sr_sad, len(y_sad))
axes[0, 0].plot(t_sad, y_sad, color='blue', linewidth=0.5)
axes[0, 0].set_title('SAD SONG - Original Waveform', fontweight='bold')
axes[0, 0].set_ylabel('Amplitude')
axes[0, 0].grid(True, alpha=0.3)
# Pitch Shifted
t_sad_pitch = np.linspace(0, len(y_sad_pitch)/sr_sad, len(y_sad_pitch))
axes[0, 1].plot(t_sad_pitch, y_sad_pitch, color='red', linewidth=0.5)
axes[0, 1].set_title('SAD SONG - Pitch-Shifted Waveform', fontweight='bold')
axes[0, 1].set_ylabel('Amplitude')
axes[0, 1].grid(True, alpha=0.3)

# HAPPY SONG
# Original
t_happy = np.linspace(0, len(y_happy)/sr_happy, len(y_happy))
axes[1, 0].plot(t_happy, y_happy, color='blue', linewidth=0.5)
axes[1, 0].set_title('HAPPY SONG - Original Waveform', fontweight='bold')
axes[1, 0].set_ylabel('Amplitude')
axes[1, 0].grid(True, alpha=0.3)
# Pitch Shifted
t_happy_pitch = np.linspace(0, len(y_happy_pitch)/sr_happy, len(y_happy_pitch))
axes[1, 1].plot(t_happy_pitch, y_happy_pitch, color='red', linewidth=0.5)
axes[1, 1].set_title('HAPPY SONG - Pitch-Shifted Waveform', fontweight='bold')
axes[1, 1].set_ylabel('Amplitude')
axes[1, 1].grid(True, alpha=0.3)

# Set xlabel for bottom plots
axes[1, 0].set_xlabel('Time (s)')
axes[1, 1].set_xlabel('Time (s)')
plt.tight_layout()
plt.show()

# Visualisasi spectrogram sebelum dan sesudah pitch shifting
fig, axes = plt.subplots(2, 2, figsize=(16, 10))

# SAD SONG
```

```
# Original
D_sad = librosa.amplitude_to_db(np.abs(librosa.stft(y_sad)), ref=np.max)
librosa.display.specshow(D_sad, sr=sr_sad, x_axis='time', y_axis='log',
                         ax=axes[0, 0], cmap='magma')
axes[0, 0].set_title('SAD SONG - Original Spectrogram', fontweight='bold')
# Pitch Shifted
D_sad_pitch = librosa.amplitude_to_db(np.abs(librosa.stft(y_sad_pitch)),
                                      ref=np.max)
librosa.display.specshow(D_sad_pitch, sr=sr_sad, x_axis='time', y_axis='log',
                         ax=axes[0, 1], cmap='magma')
axes[0, 1].set_title('SAD SONG - Pitch-Shifted Spectrogram', fontweight='bold')

# HAPPY SONG
# Original
D_happy = librosa.amplitude_to_db(np.abs(librosa.stft(y_happy)),
                                  ref=np.max)
librosa.display.specshow(D_happy, sr=sr_happy, x_axis='time', y_axis='log',
                         ax=axes[1, 0], cmap ='magma')
axes[1, 0].set_title('HAPPY SONG - Original Spectrogram', fontweight='bold')
# Pitch Shifted
D_happy_pitch = librosa.amplitude_to_db(np.abs(librosa.stft(y_happy_pitch)),
                                        ref=np.max)
librosa.display.specshow(D_happy_pitch, sr=sr_happy, x_axis='time', y_axis='log'
                         ax=axes[1, 1], cmap='magma')
axes[1, 1].set_title('HAPPY SONG - Pitch-Shifted Spectrogram', fontweight='bold'

# Set xlabel for bottom plots
axes[1, 0].set_xlabel('Time (s)')
axes[1, 1].set_xlabel('Time (s)')
plt.tight_layout()
plt.show()
```





In [35]:
```
print("SAD SONG - Original :(")
display(Audio(y_sad, rate=sr_sad))
print("SAD SONG - Pitch Shifting (5 semitone) :(")
display(Audio(y_sad_pitch, rate=sr_sad))
print("HAPPY SONG - Original :)")
display(Audio(y_happy, rate=sr_happy))
print("HAPPY SONG - Pitch Shifting (5 semitone) :)")
display(Audio(y_happy_pitch, rate=sr_happy))
```

SAD SONG - Original :(

0:00 / 1:00

SAD SONG - Pitch Shifting (5 semitone) :(

0:00 / 1:00

HAPPY SONG - Original :)

0:00 / 1:00

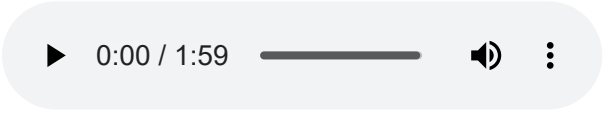
HAPPY SONG - Pitch Shifting (5 semitone) :)

0:00 / 1:00

In [36]:
```
# Crossfade
PATH_SAD = os.path.join(os.getcwd(), 'data_ho', 'lagu_1.wav')
PATH_HAPPY = os.path.join(os.getcwd(), 'data_ho', 'lagu_2.wav')
a, sr_a = librosa.load(PATH_SAD, sr=None)
b, sr_b = librosa.load(PATH_HAPPY, sr=None)

if sr_b != sr_a:
    b = librosa.resample(b, orig_sr=sr_b, target_sr=sr_a)
```

```
    sr_b = sr_a
sr = sr_a

def crossfade(a: np.ndarray, b: np.ndarray, sr: int, dur_sec: float = 1.0):
    N = min(int(dur_sec * sr), len(a), len(b))
    fade_out_a = np.linspace(1.0, 0.0, N)
    fade_in_b = np.linspace(0.0, 1.0, N)
    result = np.concatenate([a[:-N], a[-N:] * fade_out_a + b[:N] * fade_in_b, b[
    return result, fade_out_a, fade_in_b, N

y_cross, fade_out, fade_in, N = crossfade(a, b, sr, dur_sec=1.0)

waktu_mulai_a = (len(a) - N) / sr
waktu_akhir_a = len(a) / sr
print(f"Informasi Waktu Crossfade:")
print(f"Durasi crossfade: {N/sr:.2f} detik ({N:,} sampel)")
print(f"Waktu Crossfade: {waktu_mulai_a:.2f}s - {waktu_akhir_a:.2f}s")
print(f"Panjang total hasil: {len(y_cross)/sr:.2f}s")

print(f"\n🎵 Hasil Crossfade ({waktu_mulai_a:.2f}s):")
display(Audio(y_cross, rate=sr))
```

```
Informasi Waktu Crossfade:
Durasi crossfade: 1.00 detik (44,100 sampel)
Waktu Crossfade: 59.00s - 60.00s
Panjang total hasil: 119.00s

🎵 Hasil Crossfade (59.00s):
```

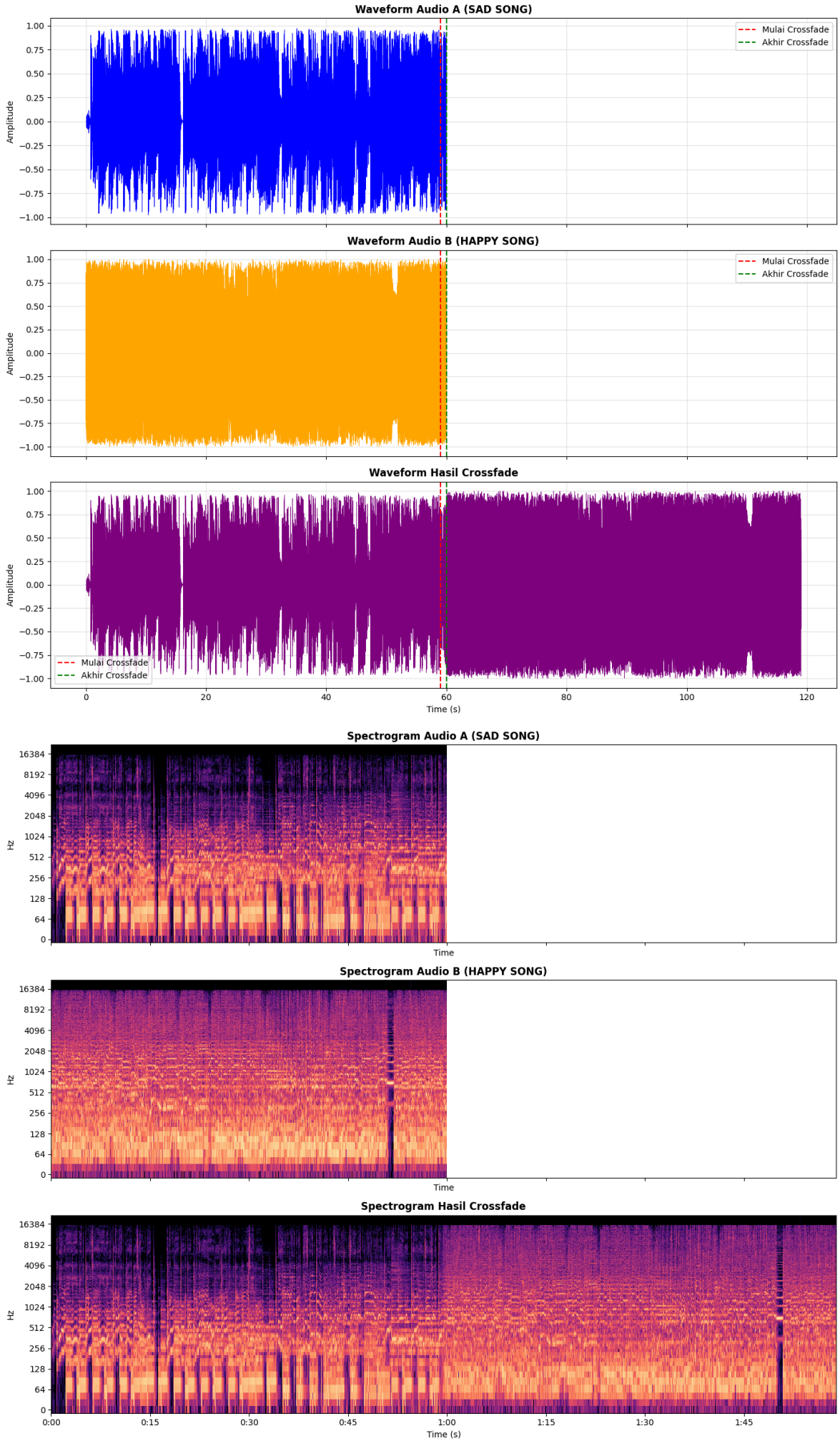


In [ ]:

```
# Visualisasi waveform dengan crossfade
t_a = np.linspace(0, len(a)/sr, len(a))
t_b = np.linspace(0, len(b)/sr, len(b))
t_cross = np.linspace(0, len(y_cross)/sr, len(y_cross))
fig, axes = plt.subplots(3, 1, figsize=(14, 12), sharex=True)

axes[0].plot(t_a, a, color='blue', linewidth=0.5)
axes[0].set_title('Waveform Audio A (SAD SONG)', fontweight='bold')
axes[0].set_ylabel('Amplitude')
axes[0].axvline(waktu_mulai_a, color='red', linestyle='--',
                label='Mulai Crossfade')
axes[0].axvline(waktu_akhir_a, color='green', linestyle='--',
                label='Akhir Crossfade')
axes[0].legend()
axes[0].grid(True, alpha=0.3)

axes[1].plot(t_b, b, color='orange', linewidth=0.5)
axes[1].set_title('Waveform Audio B (HAPPY SONG)', fontweight='bold')
axes[1].set_ylabel('Amplitude')
axes[1].axvline(waktu_mulai_a, color='red', linestyle='--',
                label='Mulai Crossfade')
axes[1].axvline(waktu_akhir_a, color='green', linestyle='--',
                label='Akhir Crossfade')
axes[1].legend()
axes[1].grid(True, alpha=0.3)

axes[2].plot(t_cross, y_cross, color='purple', linewidth=0.5)
```

```
axes[2].set_title('Waveform Hasil Crossfade', fontweight='bold')
axes[2].set_xlabel('Time (s)')
axes[2].set_ylabel('Amplitude')
axes[2].axvline(waktu_mulai_a, color='red', linestyle='--',
                label='Mulai Crossfade')
axes[2].axvline(waktu_akhir_a, color='green', linestyle='--',
                label='Akhir Crossfade')
axes[2].legend()
axes[2].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

# Visualisasi spectrogram dengan crossfade
D_a = librosa.amplitude_to_db(np.abs(librosa.stft(a)), ref=np.max)
D_b = librosa.amplitude_to_db(np.abs(librosa.stft(b)), ref=np.max)
D_cross = librosa.amplitude_to_db(np.abs(librosa.stft(y_cross)), ref=np.max)
fig, axes = plt.subplots(3, 1, figsize=(14, 12), sharex=True)

librosa.display.specshow(D_a, sr=sr, x_axis='time', y_axis='log', ax=axes[0],
                         cmap='magma')
axes[0].set_title('Spectrogram Audio A (SAD SONG)', fontweight='bold')

librosa.display.specshow(D_b, sr=sr, x_axis='time', y_axis='log', ax=axes[1],
                         cmap='magma')
axes[1].set_title('Spectrogram Audio B (HAPPY SONG)', fontweight='bold')

librosa.display.specshow(D_cross, sr=sr, x_axis='time', y_axis='log', ax=axes[2]
                         cmap='magma')
axes[2].set_title('Spectrogram Hasil Crossfade', fontweight='bold')

axes[2].set_xlabel('Time (s)')
plt.tight_layout()
plt.show()
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_3520\1918323581.py:32: UserWarning: Cr
eating legend with loc="best" can be slow with large amounts of data.
  plt.tight_layout()
```


Waveform Audio A (SAD SONG)
Mulai Crossfade
Akhir Crossfade
Amplitude
Waveform Audio B (HAPPY SONG)
Waveform Hasil Crossfade
Time (s)
Spectrogram Audio A (SAD SONG)
Hz
Time
Spectrogram Audio B (HAPPY SONG)
Spectrogram Hasil Crossfade

Penjelasan Proses dan Paramater:

1. Untuk time stretching, parameter yang digunakan yaitu strech factor = 0.5 yang membuat kedua lagu menjaid 2x lebih lambat dibandingkan durasi lagu asli. Juga rate < 1.0 yang membuat audio lebih lambat dan durasi lebih panjang.

2. Untuk pitch shifting, parameter yang digunakan yaitu 5 semitone yang membuat pitch menjadi tinggi tanpa merubah durasi audio.

3. Untuk crossfade, parameter yang digunakan yaitu 1 detik pertama dibuat fade in dan 1 detik terakhir dibuat fade out.

Penjelasan:

1. Time Stretching
   Pada waveform, SAD SONG menunjukkan amplitudo yang lebih bervariasi dan fluktuatif dengan beberapa bagian tenang di antara puncak-puncak amplitudo. Hal ini menggambarkan dinamika lagu yang lebih lembut dan emosional. Setelah dilakukan time-stretching, durasi lagu menjadi sekitar dua kali lebih panjang, amplitudo tampak lebih renggang, menandakan kecepatan lagu diperlambat tanpa mengubah tinggi nada (pitch). Sebaliknya, HAPPY SONG memiliki amplitudo yang lebih padat dan konsisten, menunjukkan energi yang lebih tinggi dan tempo yang lebih cepat. Setelah time-stretching, bentuk gelombang juga memanjang dan tampak lebih rapat secara horizontal, menandakan tempo lagu telah melambat namun tetap mempertahankan struktur aslinya.

   Pada spectrogram, baik untuk SAD SONG maupun bahagia, terlihat distribusi frekuensi yang berbeda. SAD SONG didominasi oleh energi pada frekuensi rendah hingga menengah, menunjukkan karakter nada yang lebih lembut dan mendalam. Setelah time-stretching, pola spektrum menjadi lebih panjang secara waktu, tetapi kepadatan frekuensi tetap serupa. HAPPY SONG memiliki spektrum frekuensi yang lebih luas dan intensitas energi yang lebih tinggi pada frekuensi menengah hingga tinggi, mencerminkan suasana yang ceria dan cepat. Setelah time-stretching, distribusi warna pada spectrogram melebar ke arah waktu namun tetap mempertahankan pola harmonik aslinya.

2. Pitch Shifting
   Pada waveform, baik SAD SONG maupun HAPPY SONG menunjukkan pola amplitudo yang mirip antara versi asli dan versi setelah pitch shifting. Namun, amplitudo pada hasil pitch-shifted terlihat sedikit lebih besar dan rapat, menandakan adanya peningkatan energi akibat perubahan frekuensi suara. Durasi total lagu tetap sama, karena proses pitch shifting hanya mengubah tinggi nada (frekuensi) tanpa memengaruhi kecepatan atau panjang waktu. Pada SAD SONG, bentuk gelombang tetap menunjukkan dinamika yang lembut dengan beberapa jeda, sedangkan pada HAPPY SONG, bentuk gelombang tampak lebih padat dan konstan, menandakan intensitas dan tempo yang lebih tinggi.

Pada spectrogram, perbedaan frekuensi antara versi asli dan versi pitch-shifted terlihat jelas. Pada SAD SONG, setelah dilakukan pitch shifting, komponen frekuensi tampak sedikit bergeser ke arah frekuensi yang lebih tinggi (warna-warna cerah pada pita bawah naik sedikit ke atas). Pergeseran ini menunjukkan peningkatan nada lagu. Begitu pula pada HAPPY SONG, area warna oranye dan kuning pada frekuensi menengah juga bergeser ke atas, memperlihatkan bahwa nada-nada tinggi menjadi lebih dominan. Meskipun demikian, pola energi dan distribusi temporal tetap konsisten, yang berarti struktur ritme dan tempo lagu tidak berubah.

3. Crossfade
   Pada waveform, bagian atas menunjukkan bentuk gelombang SAD SONG dengan amplitudo yang cukup bervariasi, menggambarkan dinamika lagu yang cenderung tenang dan memiliki jeda emosional. Sementara itu, HAPPY SONG memiliki amplitudo yang lebih stabil dan padat, menandakan tempo yang cepat dan energi yang konstan. Garis vertikal merah dan hijau pada kedua waveform menandakan awal dan akhir proses crossfade, yaitu saat dua audio saling bertumpang tindih. Pada hasil waveform gabungan di bagian bawah, terlihat bahwa transisi antara kedua lagu berlangsung mulus — amplitudo SAD SONG secara perlahan menurun sementara amplitudo HAPPY SONG meningkat, menunjukkan proses pencampuran audio yang halus tanpa jeda atau perubahan mendadak.

   Pada spectrogram, distribusi energi frekuensi memberikan gambaran lebih detail tentang proses transisi ini. SAD SONG memiliki intensitas warna yang lebih lembut di frekuensi rendah hingga menengah, menandakan dominasi nada rendah dan atmosfer tenang. HAPPY SONG di sisi lain menampilkan spektrum yang lebih terang dan merata di seluruh rentang frekuensi, mencerminkan karakter lagu yang ceria dan energik. Pada spectrogram hasil crossfade, terlihat perpaduan kedua karakter tersebut — bagian awal didominasi oleh pola frekuensi SAD SONG yang kemudian secara bertahap bergeser menjadi pola HAPPY SONG, dengan rentang waktu transisi yang tampak jelas namun halus di sekitar titik crossfade.

📚 REFERENSI

🔹 1. Materi Pembelajaran

Berikut referensi utama yang digunakan dalam Hands-on:

- Audio Processing Week 1
- Audio Processing Week 2
- Audio Processing Week 3

🤖 2. Bantuan AI

Proses pengerjaan notebook ini juga dibantu dengan referensi diskusi AI untuk penjelasan konsep:

- ChatGPT Session 1
- ChatGPT Session 2
- ChatGPT Session 3
