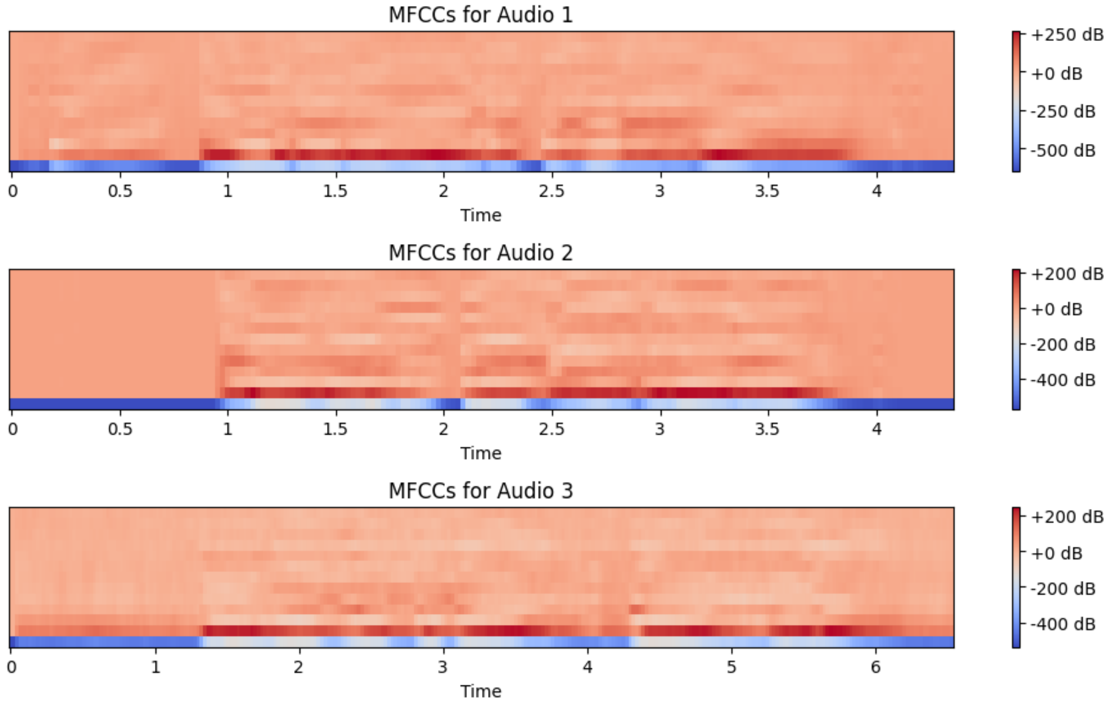
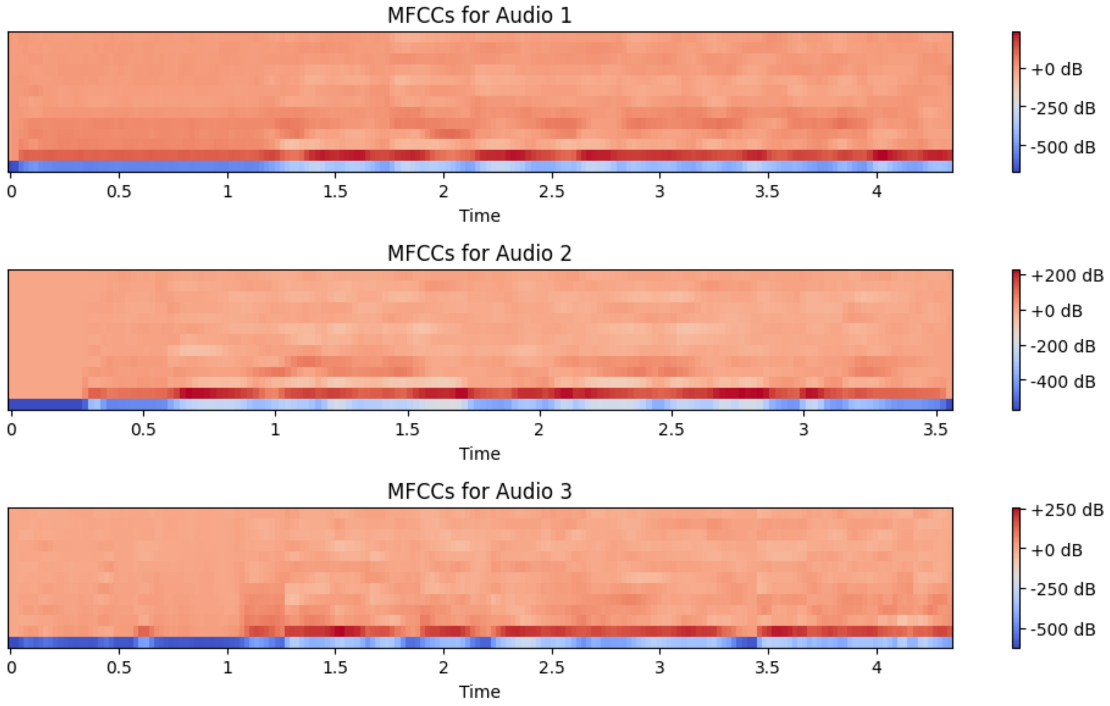
Untitled-1

May 4, 2024

[ ]:

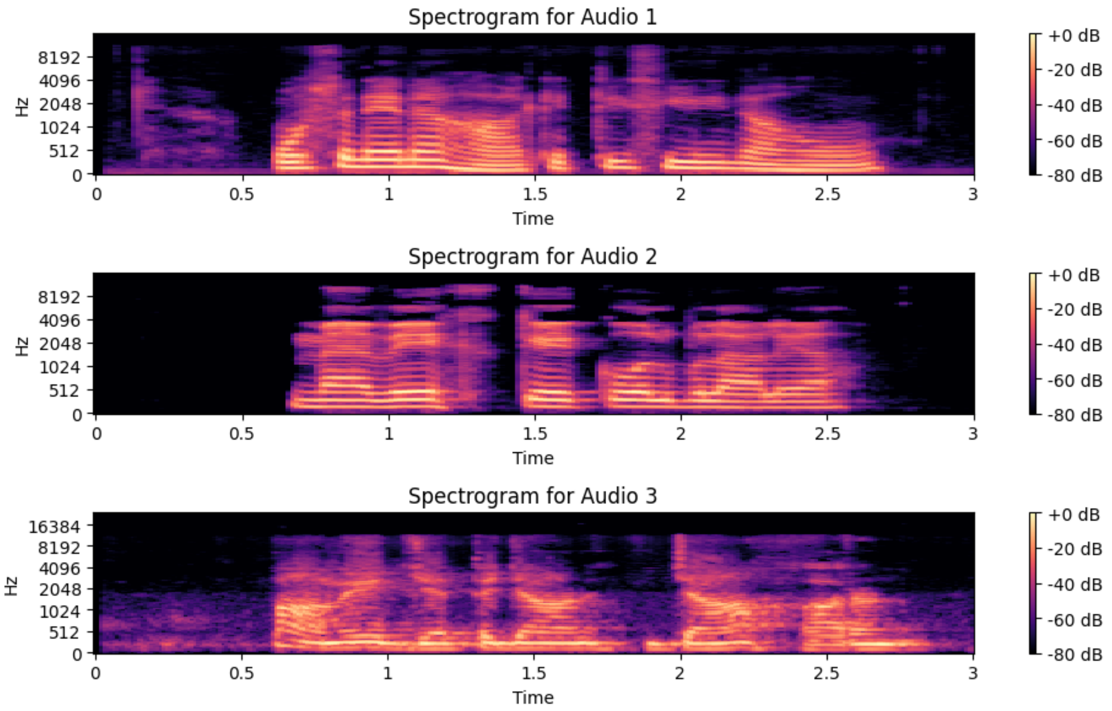
```
import os
import librosa
import librosa.display
import numpy as np
import matplotlib.pyplot as plt

# Function to load audio files from a folder
def load_audio_files(folder):
    audio_files = []
    for filename in os.listdir(folder):
        if filename.endswith('.mp3'):  # Change to '.mp3'
            file_path = os.path.join(folder, filename)
            audio_files.append(file_path)
    return audio_files

# Function to sample audio files
def sample_audio_files(audio_files, num_samples):
    return np.random.choice(audio_files, num_samples, replace=False)

# Function to extract audio features (MFCCs)
def extract_features(audio_files, num_mfcc=13):
    features = []
    for file in audio_files:
        y, sr = librosa.load(file, sr=None, duration=3)  # Limit duration to 3␣
 ↪seconds
        mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=num_mfcc)
        features.append(mfccs)
    return features

# Function to visualize audio features (MFCCs)
def visualize_features(features):
    plt.figure(figsize=(10, 6))
    for i, feature in enumerate(features):
        plt.subplot(len(features), 1, i+1)
        librosa.display.specshow(feature, x_axis='time')
        plt.colorbar(format='%+2.0f dB')
        plt.title(f'MFCCs for Audio {i+1}')
        plt.tight_layout()
    plt.show()

# Paths to Punjabi and Hindi audio folders
punjabi_folder = "/home/vansh/Desktop/cv-corpus-17.0-2024-03-15/pa-IN/clips"
hindi_folder = "/home/vansh/Desktop/cv-corpus-17.0-2024-03-15/hi/clips"

# Load audio files
punjabi_audio_files = load_audio_files(punjabi_folder)
hindi_audio_files = load_audio_files(hindi_folder)

# Sample audio files
num_samples = 3  # Number of samples to take from each language
punjabi_samples = sample_audio_files(punjabi_audio_files, num_samples)
hindi_samples = sample_audio_files(hindi_audio_files, num_samples)

# Extract audio features (MFCCs)
punjabi_features = extract_features(punjabi_samples)
hindi_features = extract_features(hindi_samples)

# Visualize audio features (MFCCs)
visualize_features(punjabi_features)
visualize_features(hindi_features)
```

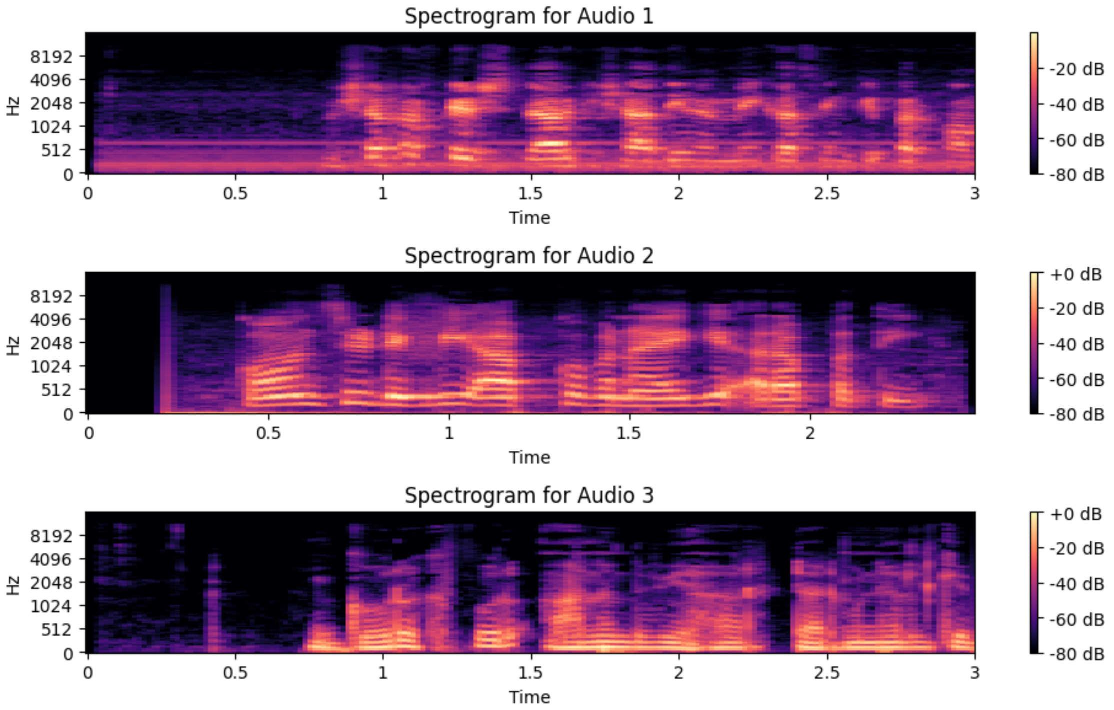




[ ]:

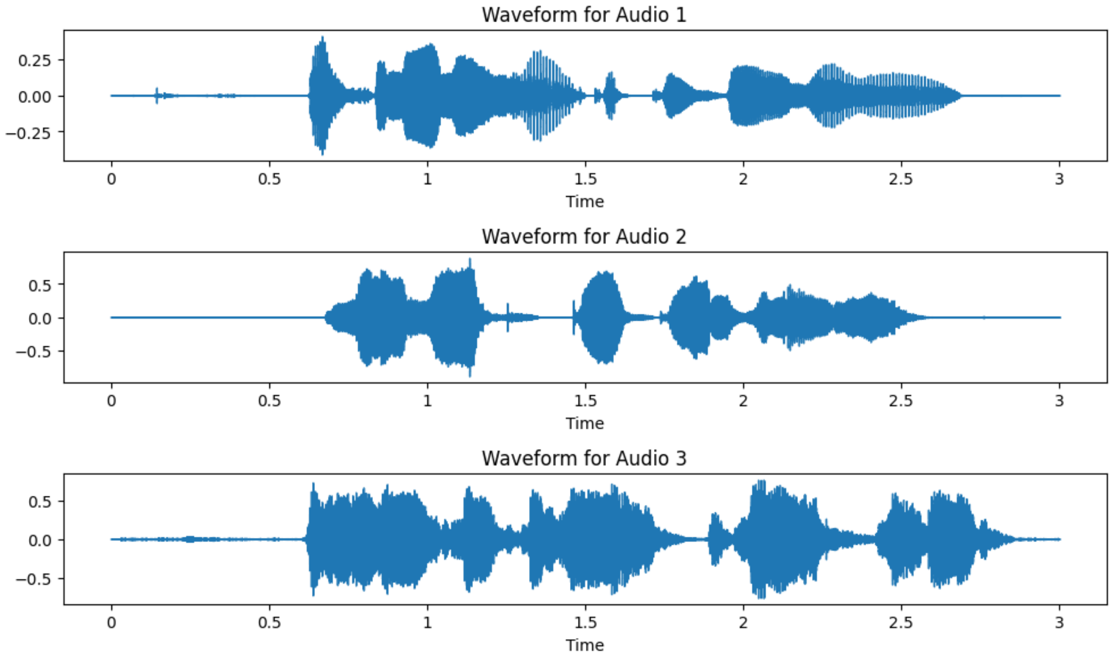
```python
def visualize_spectrogram(audio_files):
    plt.figure(figsize=(10, 6))
    for i, file in enumerate(audio_files):
        y, sr = librosa.load(file, sr=None, duration=3)
        spectrogram = librosa.feature.melspectrogram(y=y, sr=sr)
        plt.subplot(len(audio_files), 1, i+1)
        librosa.display.specshow(librosa.power_to_db(spectrogram, ref=np.max),␣
 ↪sr=sr, x_axis='time', y_axis='mel')
        plt.colorbar(format='%+2.0f dB')
        plt.title(f'Spectrogram for Audio {i+1}')
        plt.tight_layout()
    plt.show()

visualize_spectrogram(punjabi_samples)
visualize_spectrogram(hindi_samples)
```

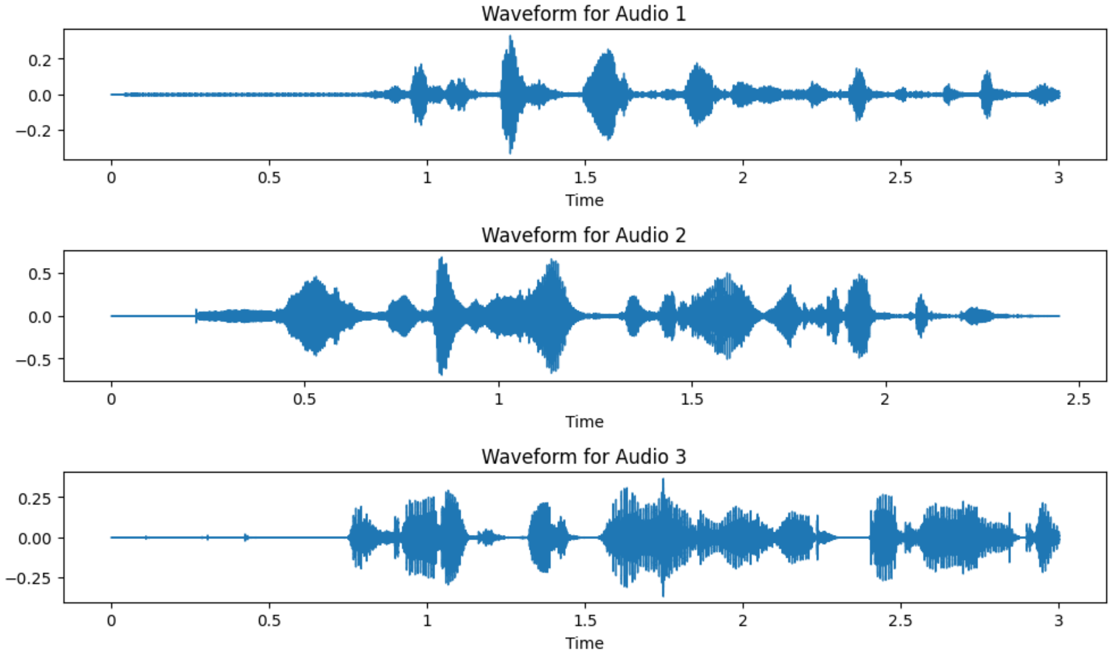

Spectrogram for Audio 1
Hz
8192
4096
2048
1024
512
0
0
0.5
1
1.5
2
2.5
3
Time
+0 dB
-20 dB
-40 dB
-60 dB
-80 dB
Spectrogram for Audio 2
Hz
8192
4096
2048
1024
512
0
0
0.5
1
1.5
2
2.5
3
Time
+0 dB
-20 dB
-40 dB
-60 dB
-80 dB
Spectrogram for Audio 3
Hz
16384
8192
4096
2048
1024
512
0
0
0.5
1
1.5
2
2.5
3
Time
+0 dB
-20 dB
-40 dB
-60 dB
-80 dB


Spectrogram for Audio 1
Hz
8192
4096
2048
1024
512
0
0
0.5
1
1.5
2
2.5
3
Time
-20 dB
-40 dB
-60 dB
-80 dB
Spectrogram for Audio 2
Hz
8192
4096
2048
1024
512
0
0
0.5
1
1.5
2
Time
+0 dB
-20 dB
-40 dB
-60 dB
-80 dB
Spectrogram for Audio 3
Hz
8192
4096
2048
1024
512
0
0
0.5
1
1.5
2
2.5
3
Time
+0 dB
-20 dB
-40 dB
-60 dB
-80 dB

[ ]:

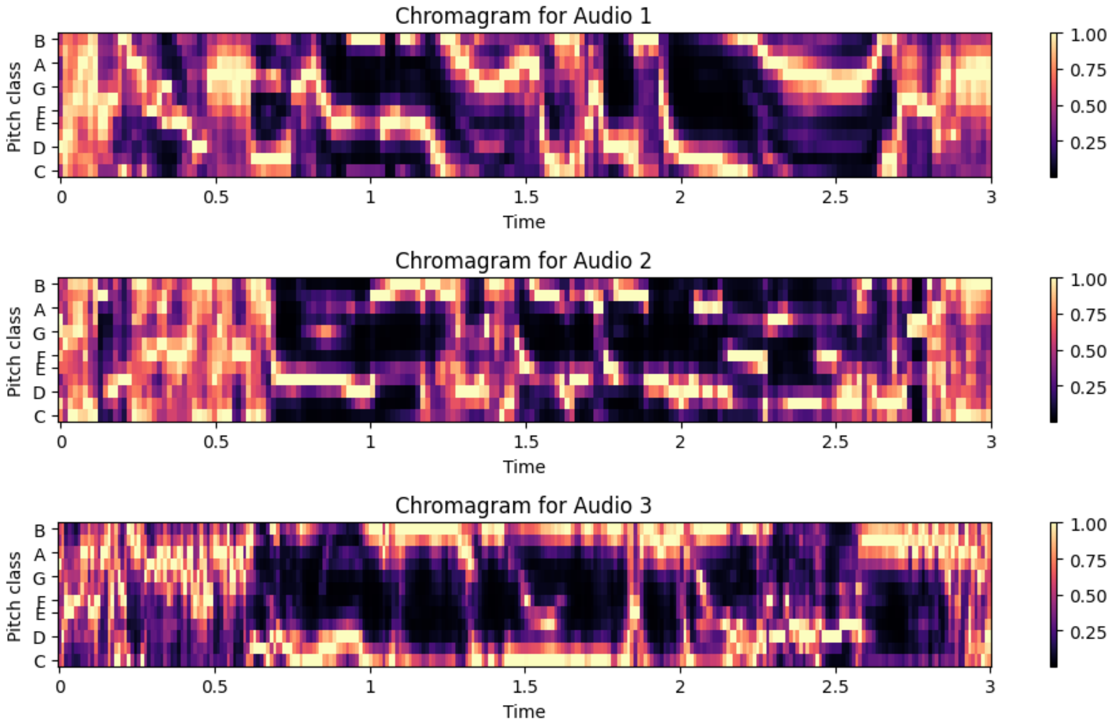
```
def visualize_waveform(audio_files):
    plt.figure(figsize=(10, 6))
    for i, file in enumerate(audio_files):
        y, sr = librosa.load(file, sr=None, duration=3)
        plt.subplot(len(audio_files), 1, i+1)
        librosa.display.waveshow(y, sr=sr)
        plt.title(f'Waveform for Audio {i+1}')
        plt.tight_layout()
    plt.show()

visualize_waveform(punjabi_samples)
visualize_waveform(hindi_samples)
```

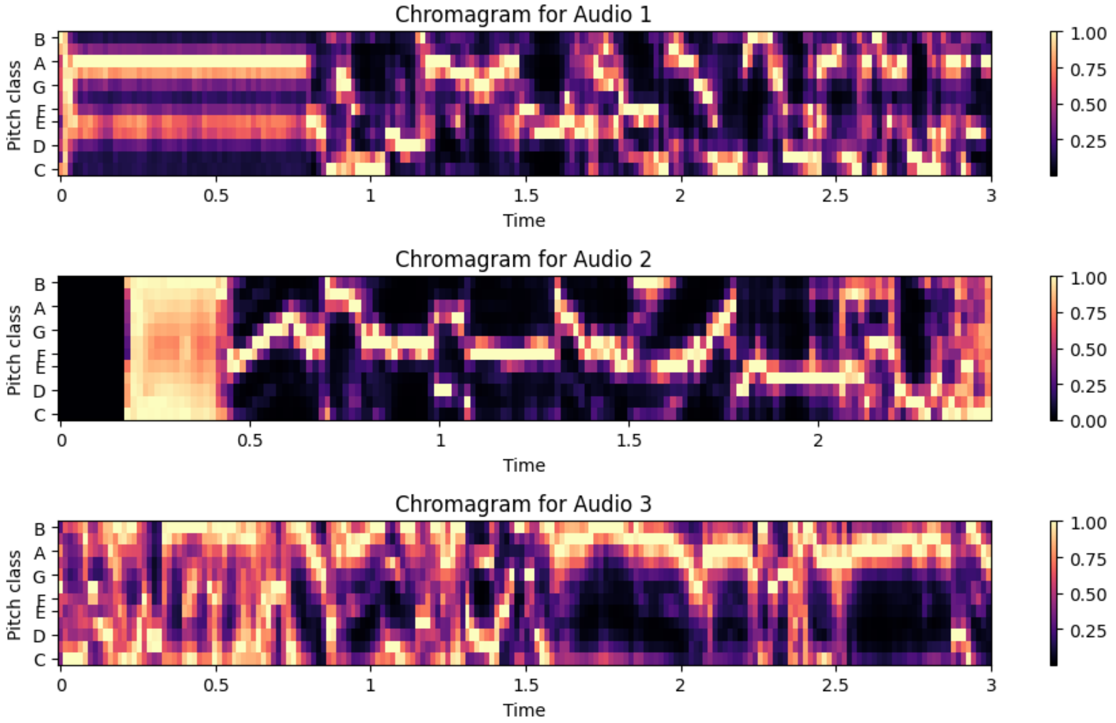




[ ]:

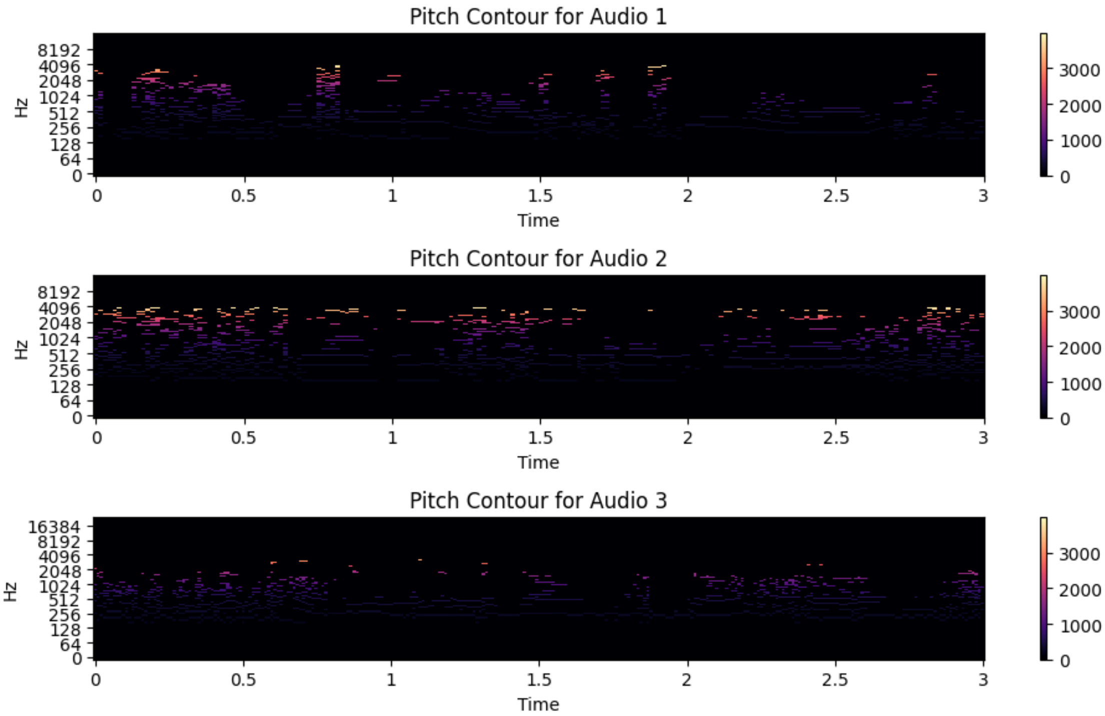
```python
def visualize_chromagram(audio_files):
    plt.figure(figsize=(10, 6))
    for i, file in enumerate(audio_files):
        y, sr = librosa.load(file, sr=None, duration=3)
        chroma = librosa.feature.chroma_stft(y=y, sr=sr)
        plt.subplot(len(audio_files), 1, i+1)
        librosa.display.specshow(chroma, sr=sr, x_axis='time', y_axis='chroma')
        plt.colorbar()
        plt.title(f'Chromagram for Audio {i+1}')
        plt.tight_layout()
    plt.show()

visualize_chromagram(punjabi_samples)
visualize_chromagram(hindi_samples)
```

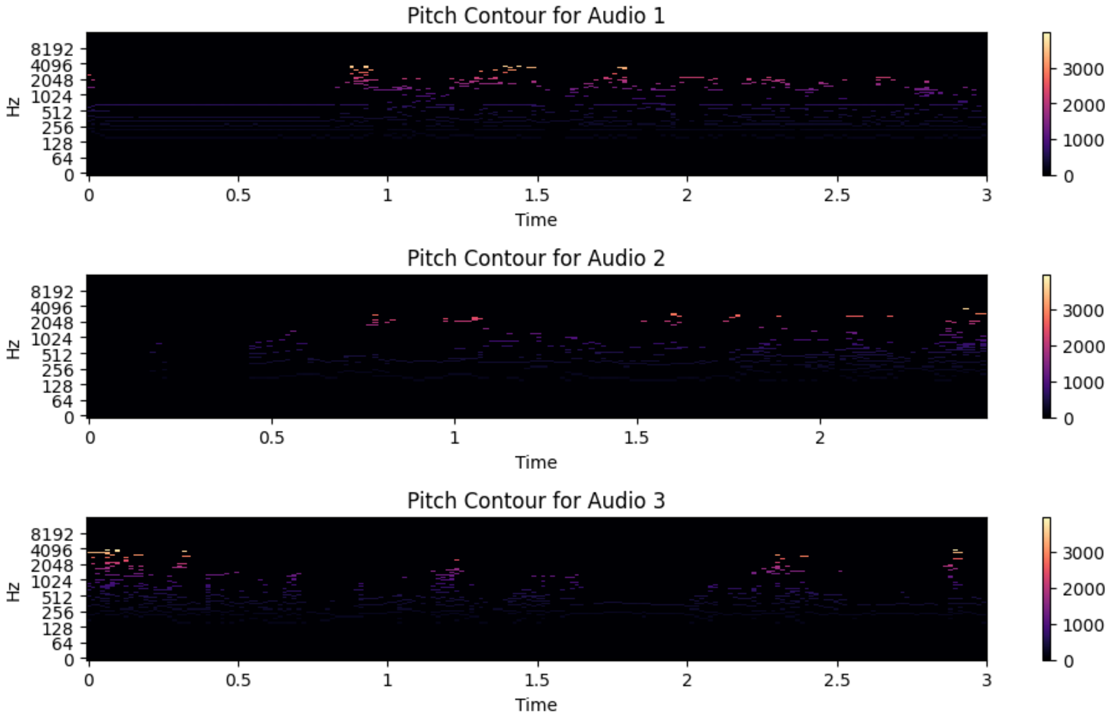

Chromagram for Audio 1
Chromagram for Audio 2
Chromagram for Audio 3
Pitch class
B
A
G
F
E
D
C
0
0.5
1
1.5
2
2.5
3
Time
1.00
0.75
0.50
0.25


Chromagram for Audio 1
Chromagram for Audio 2
Chromagram for Audio 3
Pitch class
B
A
G
F
E
D
C
0
0.5
1
1.5
2
2.5
3
Time
1.00
0.75
0.50
0.25
0.00

[ ]:

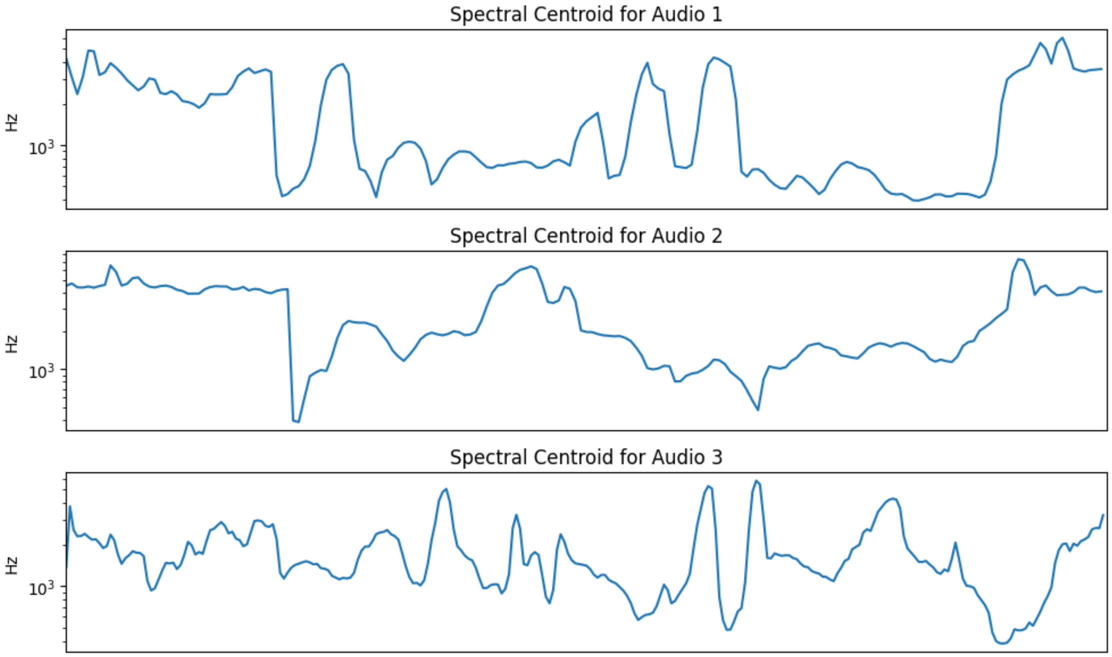
```python
def visualize_pitch_contour(audio_files):
    plt.figure(figsize=(10, 6))
    for i, file in enumerate(audio_files):
        y, sr = librosa.load(file, sr=None, duration=3)
        pitches, magnitudes = librosa.piptrack(y=y, sr=sr)
        plt.subplot(len(audio_files), 1, i+1)
        librosa.display.specshow(pitches, sr=sr, x_axis='time', y_axis='log')
        plt.colorbar()
        plt.title(f'Pitch Contour for Audio {i+1}')
        plt.tight_layout()
    plt.show()

visualize_pitch_contour(punjabi_samples)
visualize_pitch_contour(hindi_samples)
```

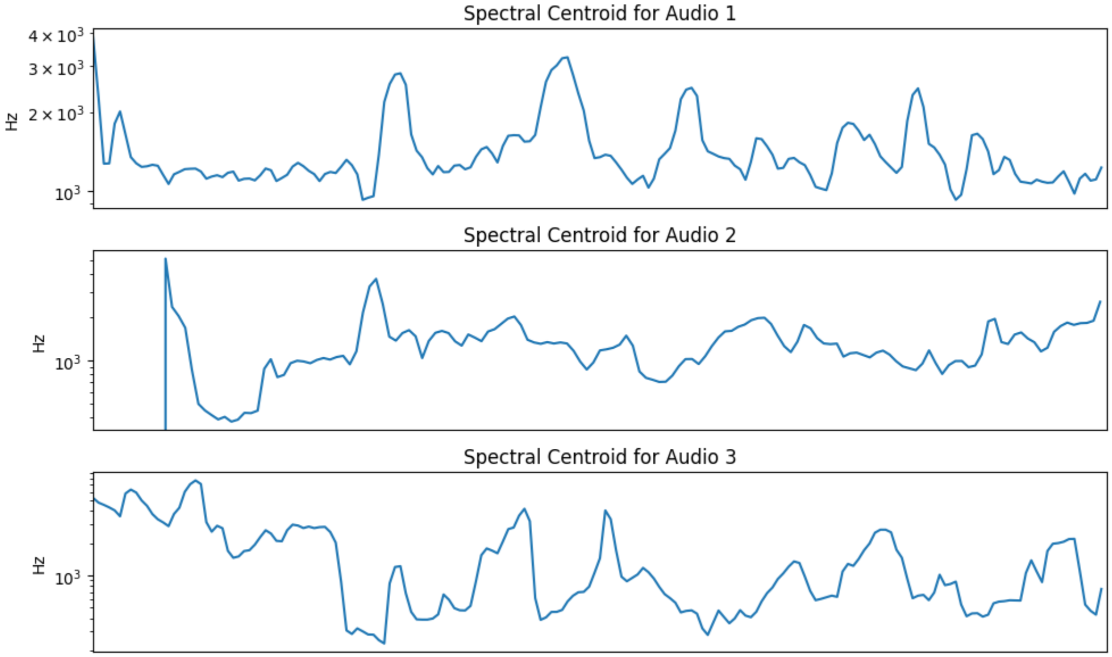




[ ]:

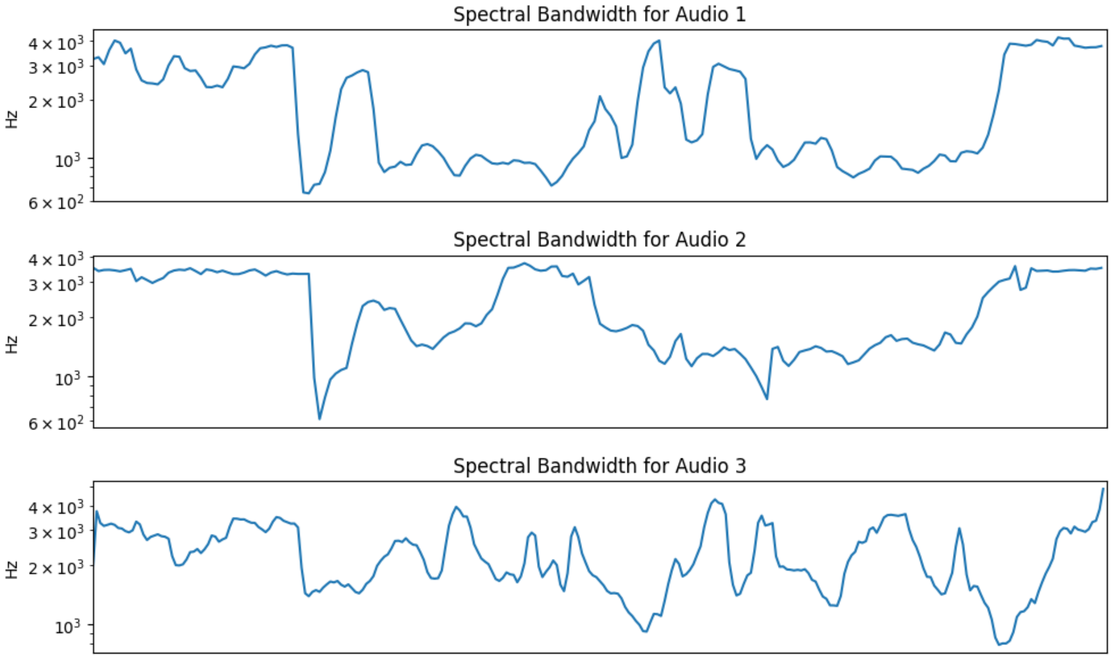
```
def visualize_spectral_centroid(audio_files):
    plt.figure(figsize=(10, 6))
    for i, file in enumerate(audio_files):
        y, sr = librosa.load(file, sr=None, duration=3)
        centroid = librosa.feature.spectral_centroid(y=y, sr=sr)
        plt.subplot(len(audio_files), 1, i+1)
        plt.semilogy(centroid.T, label='Spectral Centroid')
        plt.ylabel('Hz')
        plt.xticks([])
        plt.xlim([0, centroid.shape[-1]])
        plt.title(f'Spectral Centroid for Audio {i+1}')
        plt.tight_layout()
    plt.show()

visualize_spectral_centroid(punjabi_samples)
visualize_spectral_centroid(hindi_samples)
```





[ ]:

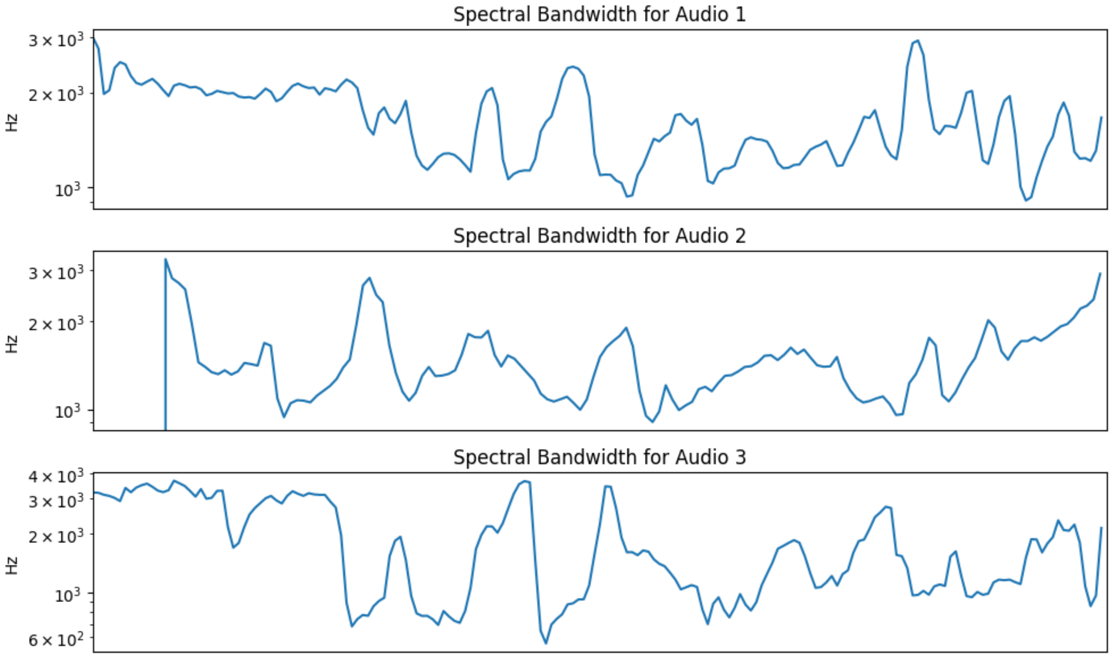
```
def visualize_spectral_bandwidth(audio_files):
    plt.figure(figsize=(10, 6))
    for i, file in enumerate(audio_files):
        y, sr = librosa.load(file, sr=None, duration=3)
        bandwidth = librosa.feature.spectral_bandwidth(y=y, sr=sr)
        plt.subplot(len(audio_files), 1, i+1)
        plt.semilogy(bandwidth.T, label='Spectral Bandwidth')
        plt.ylabel('Hz')
        plt.xticks([])
        plt.xlim([0, bandwidth.shape[-1]])
        plt.title(f'Spectral Bandwidth for Audio {i+1}')
        plt.tight_layout()
    plt.show()

visualize_spectral_bandwidth(punjabi_samples)
visualize_spectral_bandwidth(hindi_samples)
```



Spectral Bandwidth for Audio 1

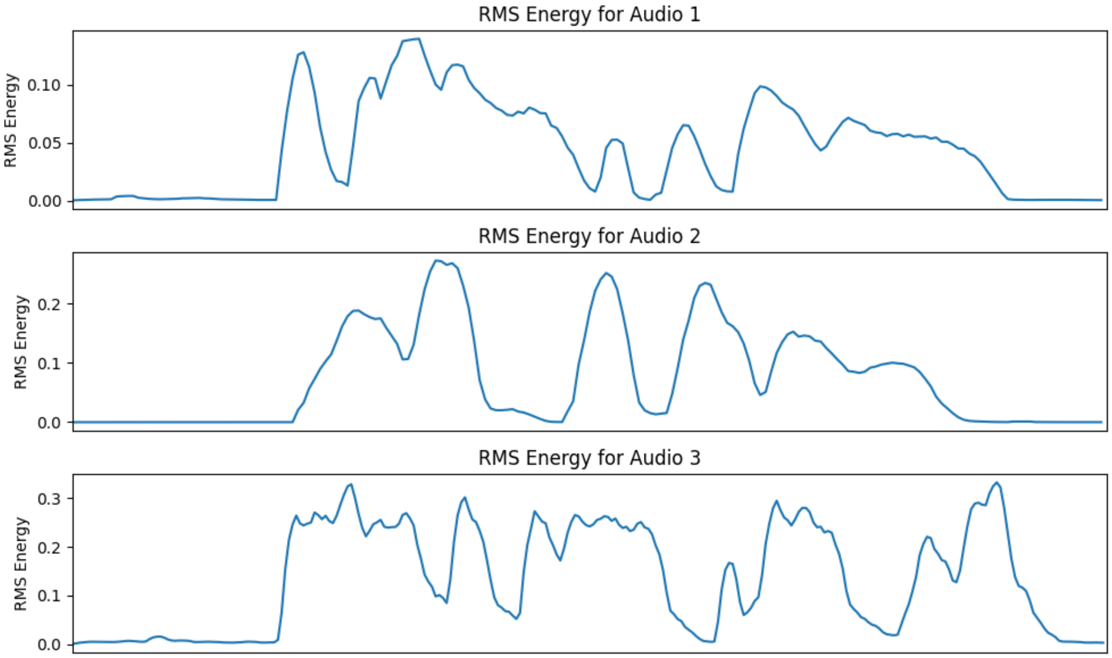
Spectral Bandwidth for Audio 2

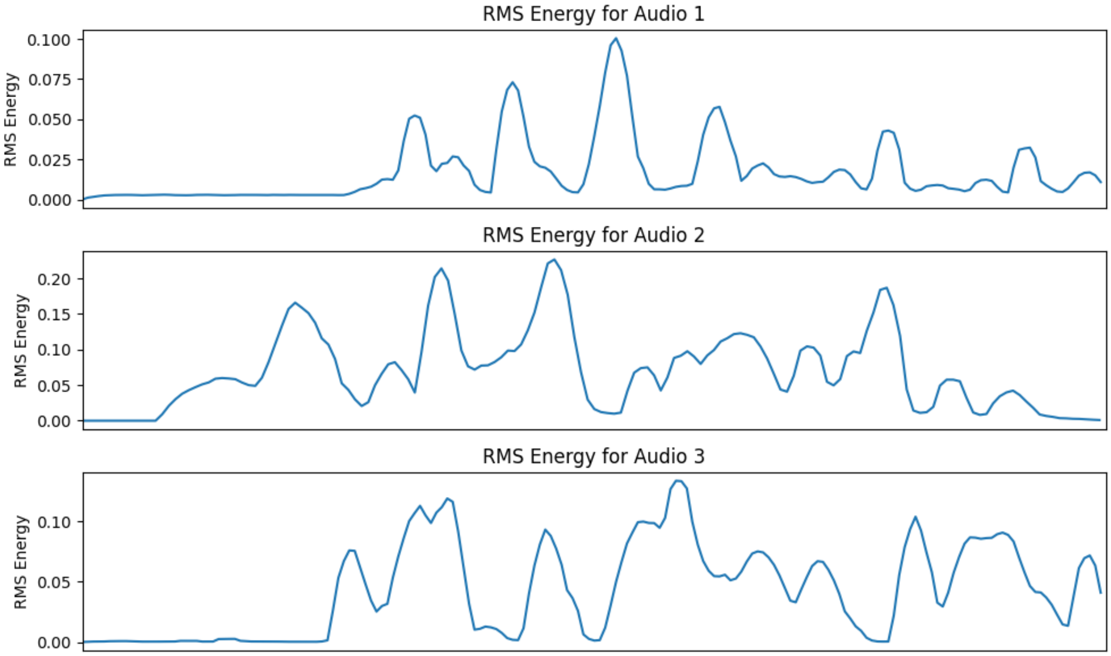
Spectral Bandwidth for Audio 3



[ ]:

```python
def visualize_rms_energy(audio_files):
    plt.figure(figsize=(10, 6))
    for i, file in enumerate(audio_files):
        y, sr = librosa.load(file, sr=None, duration=3)
        rms = librosa.feature.rms(y=y)
        plt.subplot(len(audio_files), 1, i+1)
        plt.plot(rms.T, label='RMS Energy')
        plt.ylabel('RMS Energy')
        plt.xticks([])
        plt.xlim([0, rms.shape[-1]])
        plt.title(f'RMS Energy for Audio {i+1}')
        plt.tight_layout()
    plt.show()

visualize_rms_energy(punjabi_samples)
visualize_rms_energy(hindi_samples)
```





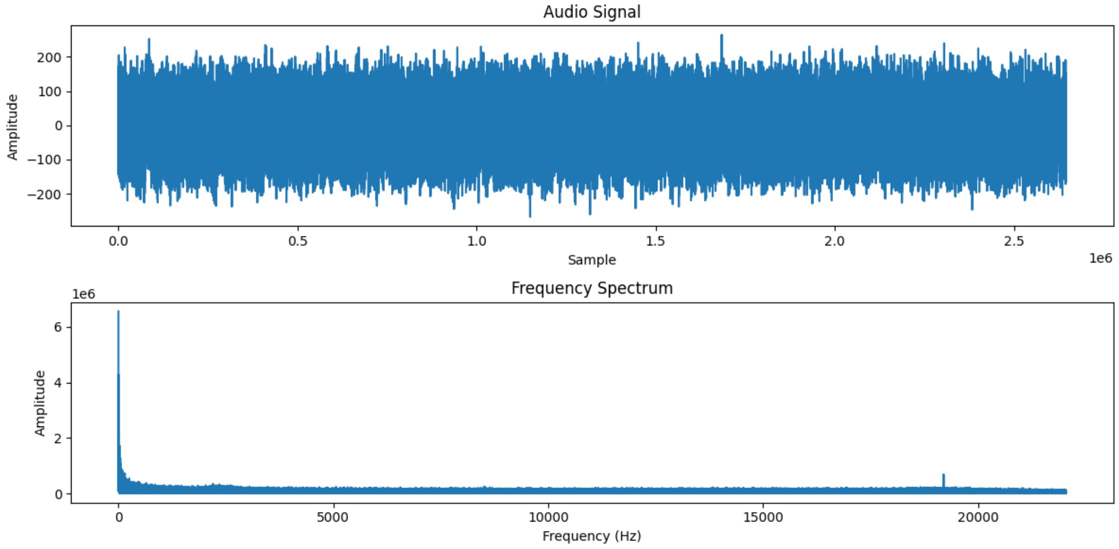
[ ]:

```
import pyaudio
import numpy as np
import matplotlib.pyplot as plt
```

```
# Constants
CHUNK_SIZE = 1024  # Number of frames per buffer
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100  # Sampling rate (Hz)
RECORD_SECONDS = 60  # Record for 60 seconds

# Function to calculate words per minute (WPM)
def calculate_wpm(audio_data):
    # Dummy implementation for demonstration
    # You need to replace this with your actual WPM calculation method
    return len(audio_data) / 1000  # Just a simple estimation

# Initialize PyAudio
p = pyaudio.PyAudio()

# Open a stream for audio input
stream = p.open(format=FORMAT,
                channels=CHANNELS,
                rate=RATE,
                input=True,
                frames_per_buffer=CHUNK_SIZE)

print("Recording...")

# Initialize an empty list to store recorded audio
audio_frames = []

# Record audio for RECORD_SECONDS
for _ in range(0, int(RATE / CHUNK_SIZE * RECORD_SECONDS)):
    data = stream.read(CHUNK_SIZE)
    audio_frames.append(np.frombuffer(data, dtype=np.int16))

print("Finished recording.")

# Stop and close the stream
stream.stop_stream()
stream.close()
p.terminate()

# Concatenate audio frames into a single array
audio_data = np.concatenate(audio_frames)

# Calculate amplitude
amplitude = np.abs(audio_data).max()

# Perform frequency analysis using FFT
```

```
freq_data = np.fft.fft(audio_data)
freqs = np.fft.fftfreq(len(freq_data), 1/RATE)
# Calculate words per minute (WPM)
wpm = calculate_wpm(audio_data)

# Plot the audio signal
plt.figure(figsize=(12, 6))

# Plot audio signal
plt.subplot(2, 1, 1)
plt.plot(audio_data)
plt.title('Audio Signal')
plt.xlabel('Sample')
plt.ylabel('Amplitude')

# Plot frequency spectrum
plt.subplot(2, 1, 2)
plt.plot(freqs[:len(freq_data)//2], np.abs(freq_data[:len(freq_data)//2]))
plt.title('Frequency Spectrum')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude')

plt.tight_layout()
plt.show()

# Display audio properties
print(f"Amplitude: {amplitude}")
print(f"Words Per Minute (WPM): {wpm}")
```

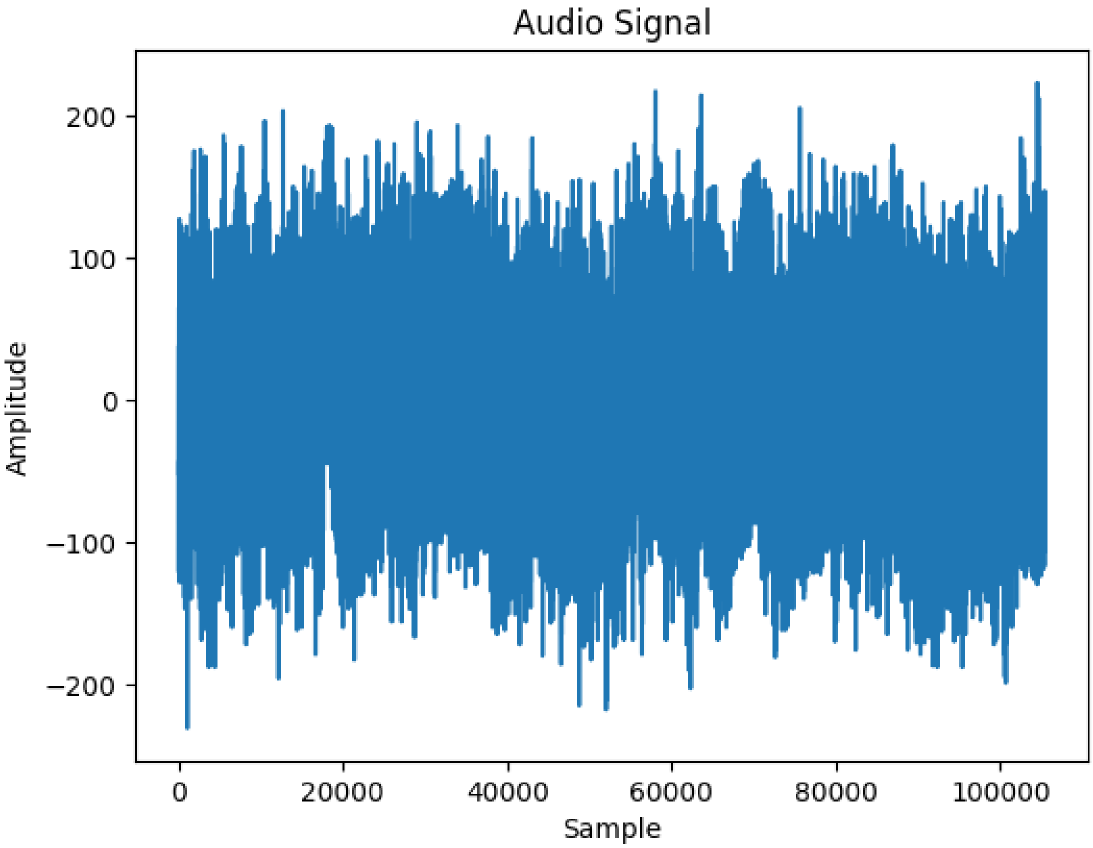
```
Recording…
Finished recording.
```



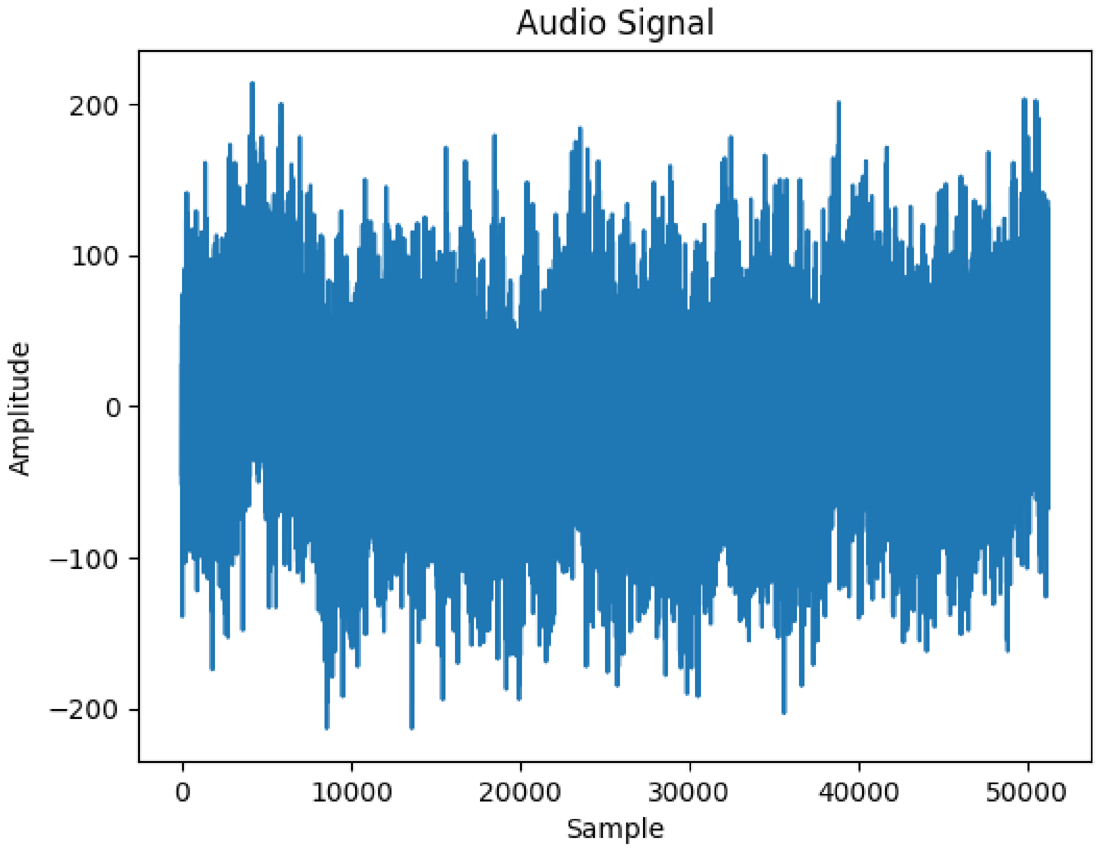
```
Amplitude: 268
Peak Frequency: 0.0 Hz
Words Per Minute (WPM): 2644.992
```

[ ]:

```python
import sys
import numpy as np
import pyaudio
from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton,␣
  ↪QVBoxLayout, QWidget
from PyQt5.QtCore import QTimer
import matplotlib.pyplot as plt

# Constants
CHUNK_SIZE = 1024  # Number of frames per buffer
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100  # Sampling rate (Hz)

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Audio Recorder")
        self.setGeometry(100, 100, 800, 600)

        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)

        self.layout = QVBoxLayout()
        self.central_widget.setLayout(self.layout)

        self.start_button = QPushButton("Start Recording")
        self.start_button.clicked.connect(self.start_recording)
        self.layout.addWidget(self.start_button)

        self.stop_button = QPushButton("Stop Recording")
        self.stop_button.clicked.connect(self.stop_recording)
        self.stop_button.setEnabled(False)
        self.layout.addWidget(self.stop_button)

        self.audio_data = []
        self.audio_stream = None
        self.timer = QTimer()
        self.timer.timeout.connect(self.update_plot)
    def start_recording(self):
        self.start_button.setEnabled(False)
        self.stop_button.setEnabled(True)
        self.audio_data = []
        self.audio_stream = pyaudio.PyAudio().open(format=FORMAT,
                                                   channels=CHANNELS,
                                                   rate=RATE,
                                                   input=True,
                                                   frames_per_buffer=CHUNK_SIZE)
        self.timer.start(100)  # Update plot every 100 milliseconds

    def stop_recording(self):
        self.start_button.setEnabled(True)
        self.stop_button.setEnabled(False)
        self.timer.stop()
        self.audio_stream.stop_stream()
        self.audio_stream.close()
        self.process_audio()

    def update_plot(self):
        data = self.audio_stream.read(CHUNK_SIZE)
        self.audio_data.extend(np.frombuffer(data, dtype=np.int16))

    def process_audio(self):
        # Plot audio signal
        plt.figure()
        plt.plot(self.audio_data)
        plt.title('Audio Signal')
        plt.xlabel('Sample')
        plt.ylabel('Amplitude')
        plt.show()

        # Calculate and display audio properties
        amplitude = np.abs(self.audio_data).max()
        peak_freq = np.fft.fft(self.audio_data).argmax() * (RATE / len(self.
 ↪audio_data))
        wpm = len(self.audio_data) / (RATE * 60)
        print(f"Amplitude: {amplitude}")
        print(f"Peak Frequency: {peak_freq} Hz")
        print(f"Words Per Minute (WPM): {wpm}")


if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```



```
Amplitude: 231
Peak Frequency: 5.017445388349515 Hz
Words Per Minute (WPM): 0.03986092214663643
```



```
Amplitude: 214
Peak Frequency: 0.861328125 Hz
Words Per Minute (WPM): 0.019349962207105064
```

```
An exception has occurred, use %tb to see the full traceback.

SystemExit: 0
```

```
/home/vansh/.local/lib/python3.10/site-
packages/IPython/core/interactiveshell.py:3561: UserWarning: To exit: use
'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```