In [50]:
```
import os
import librosa
import librosa.display
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import soundfile as sf
import speech_recognition as sr
import nltk
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import IPython.display as ipd
from IPython.display import Markdown, display

#nltk.download('punkt')
```

In [51]:
```
def load_audio(file_path, sr=16000):
    audio, _ = librosa.load(file_path, sr=sr)
    return audio
```

In [52]:
```
## 🎧 2. Load and Play Sample Audio
AUDIO_DIR = 'audio_samples'  # Replace with your directory
audio_files = [os.path.join(AUDIO_DIR, f) for f in os.listdir(AUDIO_DIR) if
print(f"Found {len(audio_files)} audio files.")

# Play one sample
ipd.Audio(audio_files[0])
```

```
Found 11 audio files.
```

Out[52]:

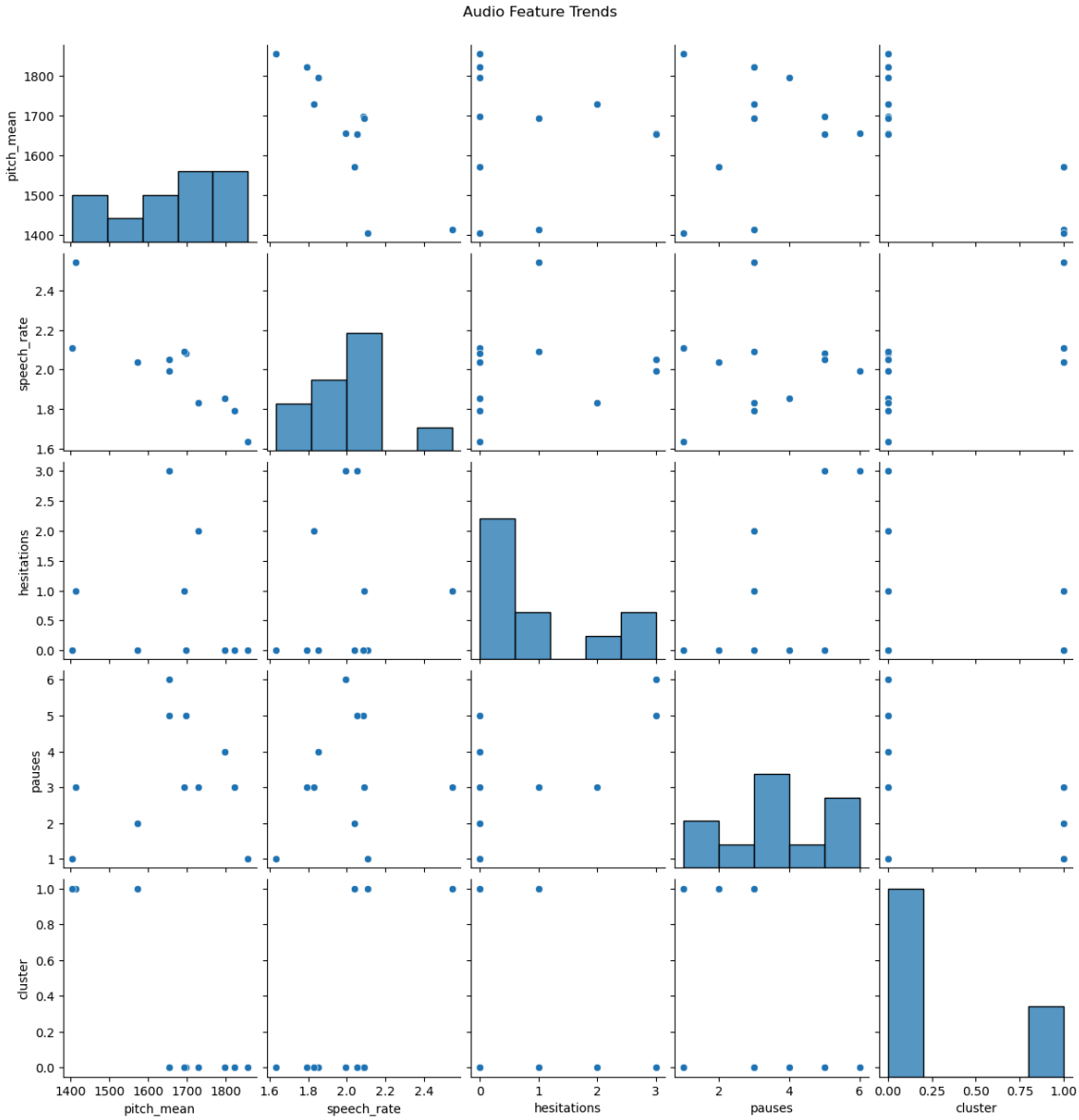
0:00 / 0:05

In [53]:
```
#Translatee Audio to Text
recognizer = sr.Recognizer()

def transcribe_audio(file_path):
    with sr.AudioFile(file_path) as source:
        audio = recognizer.record(source)
    try:
        text = recognizer.recognize_google(audio)
    except sr.UnknownValueError:
        text = ""
    return text

transcriptions = [transcribe_audio(f) for f in audio_files]
```

In [48]:
```
def extract_features(file_path, transcript):
    audio, sr = librosa.load(file_path, sr=16000)

    # Pitch
    pitches, magnitudes = librosa.piptrack(y=audio, sr=sr)
    pitch = np.mean(pitches[pitches > 0])

    # Speech rate
    words = nltk.word_tokenize(transcript)
    duration = librosa.get_duration(y=audio, sr=sr)
    speech_rate = len(words) / duration if duration else 0

    # Hesitation markers
    hesitation_count = sum([transcript.lower().count(h) for h in ['uh', 'um

    # Pauses
    intervals = librosa.effects.split(audio, top_db=25)
    silence_durations = [((j - i) / sr) for (i, j) in intervals]
    pauses = len(silence_durations)

    return {
        "pitch_mean": pitch,
        "speech_rate": speech_rate,
        "hesitations": hesitation_count,
        "pauses": pauses
    }

features = [extract_features(file, trans) for file, trans in zip(audio_file
df_features = pd.DataFrame(features)
df_features.head()
X = df_features.copy()
```

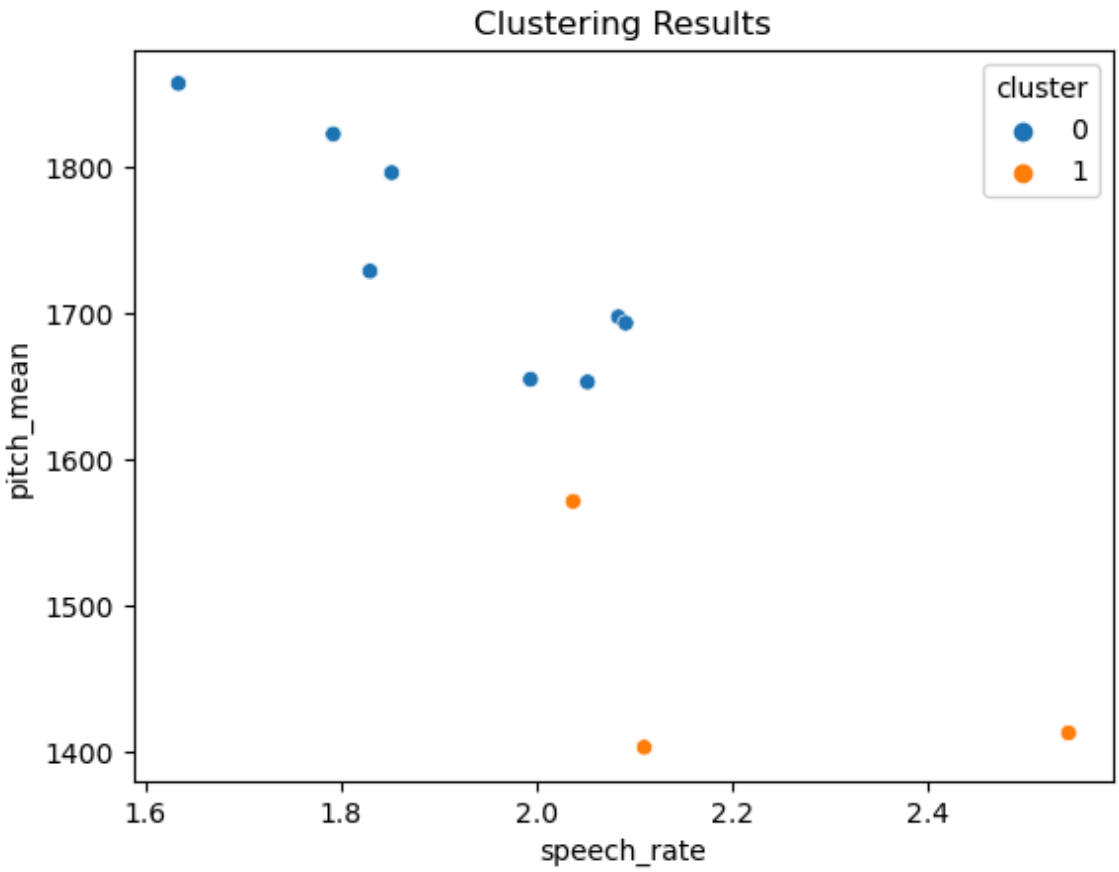
In [65]:
```
df_features['cluster'] = clusters
sns.pairplot(df_features)

plt.suptitle("Audio Feature Trends", y=1.02)
plt.show()
```

```
C:\Users\91709\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWa
rning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [58]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_features)

kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(X_scaled)
df_features['cluster'] = clusters

sns.scatterplot(x="speech_rate", y="pitch_mean", hue="cluster", data=df_fea
plt.title("Clustering Results")
plt.show()
```

```
C:\Users\91709\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\91709\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MKL,
when there are less chunks than available threads. You can avoid it by set
ting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [67]:
```
## 📝 7. Final Report Summary
def generate_report(features, centers):
    report = f"""
### 📝 Final Report Summary

**Insightful Features** used in modeling:
- {', '.join(features)}

**Clustering Centers (Scaled)**:
```
{centers}
```


"""
    display(Markdown(report))

generate_report(X.columns.tolist(), kmeans.cluster_centers_)

## 💉 8. API-Ready Risk Prediction Function
def predict_risk_from_audio(file_path, scaler, kmeans_model):
    transcript = transcribe_audio(file_path)
    features = extract_features(file_path, transcript)

    feature_vector = np.array([features[col] for col in X.columns])
    feature_scaled = scaler.transform([feature_vector])

    cluster = kmeans_model.predict(feature_scaled)[0]
    risk_score = 1 if cluster == 1 else 0  # Assuming cluster 1 is high-ris

    return {
        "transcript": transcript,
        "features": features,
        "risk_score": risk_score,
        "cluster": cluster
    }

# Example usage:
audio_path = audio_files[0]
result = predict_risk_from_audio(audio_path, scaler, kmeans)
print(f"Risk Score: {result['risk_score']} (Cluster: {result['cluster']})")
```

## 📝 Final Report Summary

**Insightful Features** used in modeling:

- pitch_mean, speech_rate, hesitations, pauses

**Clustering Centers (Scaled)**:

```
[[ 0.52413838 -0.38152158  0.18545634  0.30935922]
 [-1.39770234  1.01739087 -0.49455025 -0.82495791]]
```

```
Risk Score: 0 (Cluster: 0)
```

```
C:\Users\91709\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarnin
g: X does not have valid feature names, but StandardScaler was fitted with
feature names
  warnings.warn(
```

In [64]:
```
#Replace with the actual path to your new audio file
new_audio_path = 'Intern Video.wav' #example one

# Run the risk prediction
new_result = predict_risk_from_audio(new_audio_path, scaler, kmeans)

# Print results
print("Transcript:\n", new_result['transcript'])
print("\nExtracted Features:\n", new_result['features'])
print(f"\n🧠 Predicted Risk Score: {new_result['risk_score']} (Cluster: {ne
```

```
Transcript:
 hi my name is Sharad Babu let me explain you the clear process of the dat
a processing and PDA briefly introducing the data processing or delivery p
rocessing works like understanding the raw data the common issues in raw d
ata are like missing values for inconsistent data you are going to get a m
any outlet handling that missing data remaining the duplicates and inconsi
stency what are the possibilities to use features caring and transformatio
ns everything comes under apart from that next exploratory data analysis t
o describe the data you know better detection relations between the two va
riables like feature variable something we can describe with the visualisa
tion of the categorical variables and numerical variables we can use box w
e can easily get it from the Indian is also an essential as your business
it will be ok understanding SQL is a necessary for data analysis from the
basic of selecting the data like using the select command and determine th
e data from the database creating the databases aggregating the data like
the data number of rows are you need some average maximum minimum maximum
maximum data from the databases by writing explanation thank you

Extracted Features:
 {'pitch_mean': 1441.8832, 'speech_rate': 1.1077351336420738, 'hesitation
s': 16, 'pauses': 152}

🧠 Predicted Risk Score: 0 (Cluster: 0)

C:\Users\91709\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarnin
g: X does not have valid feature names, but StandardScaler was fitted with
feature names
  warnings.warn(
```

In [ ]: