In [1]:
```
# This file is a modified version of the original:
# https://www.kaggle.com/code/burhanuddinlatsaheb/eda-visualizations-audio-exploration
```

EDA| Visualizations + Audio Exploration

Created By Burhanuddin Latsaheb

EDA| Visualizations + Audio Exploration

If you find this notebook useful,support with an upvote👍👍

INTRODUCTION

Notebook Overview :

1. An extensive **EDA** of the different Bird sounds and its different characteristics
2. The goal of this competition is to use machine learning to **identify Eastern African bird species by sound**.

Introduction to the BirdCLEF 2023 Competition:

In this competition, participants are challenged to develop algorithms that can accurately identify bird species from audio recordings. The dataset consists of over 1 million audio recordings from around the world, with a total duration of over 1,000 hours. Each recording is labeled with the corresponding bird species, and participants are tasked with developing a machine learning model that can accurately classify the species in new, unseen recordings.

The BirdCLEF 2023 competition provides a unique opportunity for researchers and data scientists to advance the field of bioacoustics and contribute to the preservation of bird populations around the world.

TABLE OF CONTENTS

1. IMPORTS📂

Top ↑

In [2]:
```
import ast
import librosa
import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
from collections import Counter
import matplotlib.pyplot as plt
from IPython.display import Audio
```

2. 📊EDA 📊

Top ↑

Observations in Train Meta Data:

* *There are total of **12** columns and **16941** rows in **train meta** data.*
* *Train data contains **16941** values with **454(0.2%)** of missing values.*
* There are total of **12** columns : **3** numerical , **9** categorical
* ***454 Missing** values in the **train meta** data.*

In [28]:
```
trainmeta_df = pd.read_csv("/kaggle/input/birdclef-2023/train_metadata.csv")
trainmeta_df.head()
```

Out[28]:

	primary_label	secondary_labels	type	latitude	longitude	scientific_name	common_name	author	license	rating	url	
0	abethr1	[]	['song']	4.3906	38.2788	Turdus tephronotus	African Bare-eyed Thrush	Rolf A. de By	Creative Commons Attribution-NonCommercial-Sha...	4.0	https://www.xeno-canto.org/128013	abethr1/XC128
1	abethr1	[]	['call']	-2.9524	38.2921	Turdus tephronotus	African Bare-eyed Thrush	James Bradley	Creative Commons Attribution-NonCommercial-Sha...	3.5	https://www.xeno-canto.org/363501	abethr1/XC363
2	abethr1	[]	['song']	-2.9524	38.2921	Turdus tephronotus	African Bare-eyed Thrush	James Bradley	Creative Commons Attribution-NonCommercial-Sha...	3.5	https://www.xeno-canto.org/363502	abethr1/XC363
3	abethr1	[]	['song']	-2.9524	38.2921	Turdus tephronotus	African Bare-eyed Thrush	James Bradley	Creative Commons Attribution-NonCommercial-Sha...	5.0	https://www.xeno-canto.org/363503	abethr1/XC363
4	abethr1	[]	['call', 'song']	-2.9524	38.2921	Turdus tephronotus	African Bare-eyed Thrush	James Bradley	Creative Commons Attribution-NonCommercial-Sha...	4.5	https://www.xeno-canto.org/363504	abethr1/XC363

In [4]:
```
x = trainmeta_df[trainmeta_df["primary_label"] == "afgfly1"]
x
```

Out[4]:

	primary_label	secondary_labels	type	latitude	longitude	scientific_name	common_name	author	license	rating	url	
356	afgfly1	[]	['song']	-7.5253	34.8521	Bradornis microrhynchus	African Gray Flycatcher	Martin St-Michel	Creative Commons Attribution-NonCommercial-Sha...	3.0	https://www.xeno-canto.org/134487	afgfly
357	afgfly1	[]	['call']	8.9200	40.0430	Bradornis microrhynchus	African Gray Flycatcher	Andrew Spencer	Creative Commons Attribution-NonCommercial-Sha...	5.0	https://www.xeno-canto.org/267773	afgfly
358	afgfly1	[]	['call']	8.9200	40.0430	Bradornis microrhynchus	African Gray Flycatcher	Andrew Spencer	Creative Commons Attribution-NonCommercial-Sha...	5.0	https://www.xeno-canto.org/267774	afgfly
359	afgfly1	[]	['call']	-1.5765	36.6316	Bradornis microrhynchus	African Gray Flycatcher	James Bradley	Creative Commons Attribution-NonCommercial-Sha...	5.0	https://www.xeno-canto.org/344741	afgfly
360	afgfly1	['combul2', 'kerspa2', 'ratcis1']	['call,begging call,juvenile']	-2.8145	37.4113	Bradornis microrhynchus	African Gray Flycatcher	Rory Nefdt	Creative Commons Attribution-NonCommercial-Sha...	3.5	https://www.xeno-canto.org/397762	afgfly
361	afgfly1	[]	['adult', 'call', 'sex uncertain']	-3.1481	36.6951	Bradornis microrhynchus	African Gray Flycatcher	isaac kilusu	Creative Commons Attribution-NonCommercial-Sha...	4.0	https://www.xeno-canto.org/609474	afgfly
362	afgfly1	[]	['call', 'juvenile', 'sex uncertain']	-3.1481	36.6951	Bradornis microrhynchus	African Gray Flycatcher	isaac kilusu	Creative Commons Attribution-NonCommercial-Sha...	5.0	https://www.xeno-canto.org/609492	afgfly
363	afgfly1	[]	['adult', 'call', 'sex uncertain']	-3.1481	36.6951	Bradornis microrhynchus	African Gray Flycatcher	isaac kilusu	Creative Commons Attribution-NonCommercial-Sha...	5.0	https://www.xeno-canto.org/609493	afgfly

Train Meta Data Info

Top ↑

🙌 **Further Insights of train dataset**

* ***Latitude*** & ***Longitude*** *have* ***227(1.34%) Missing Values***
* ***Longitude*** & ***Rating*** *are* ***skewed***
* ***Primary labels*** , ***Secondary Labels*** , ***Type*** , ***Scientific Name*** , ***Common Name*** , ***Author*** ,***Filename*** *have* ***high Cardinality***

In [5]:
```
print(trainmeta_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16941 entries, 0 to 16940
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   primary_label     16941 non-null  object 
 1   secondary_labels  16941 non-null  object 
 2   type              16941 non-null  object 
 3   latitude          16714 non-null  float64
 4   longitude         16714 non-null  float64
 5   scientific_name   16941 non-null  object 
 6   common_name       16941 non-null  object 
 7   author            16941 non-null  object 
 8   license           16941 non-null  object 
 9   rating            16941 non-null  float64
 10  url               16941 non-null  object 
 11  filename          16941 non-null  object 
dtypes: float64(3), object(9)
memory usage: 1.6+ MB
None
```

Duplicate values in train metadata

In [30]:
```
ool_idxs = []

for col in trainmeta_df.columns:
    col_list = []
    col_list.append(col)
    if col != "url": col_list.append("url")
    if col != "filename": col_list.append("filename")
    train_metadata_dup = trainmeta_df.drop(columns = col_list)
    dup_idx = list(train_metadata_dup.loc[train_metadata_dup.duplicated(keep = "first")].index)
    print(f"# of duplicates: {len(dup_idx)} --> columns dropped: {col_list}")

    if col == "primary_label":
        ool_idxs.append(dup_idx)

trainmeta_df.loc[sum(ool_idxs, [])]["primary_label"].value_counts()
```

```
# of duplicates: 2503 --> columns dropped: ['primary_label', 'url', 'filename']
# of duplicates: 2681 --> columns dropped: ['secondary_labels', 'url', 'filename']
# of duplicates: 3428 --> columns dropped: ['type', 'url', 'filename']
# of duplicates: 2510 --> columns dropped: ['latitude', 'url', 'filename']
# of duplicates: 2512 --> columns dropped: ['longitude', 'url', 'filename']
# of duplicates: 2503 --> columns dropped: ['scientific_name', 'url', 'filename']
# of duplicates: 2503 --> columns dropped: ['common_name', 'url', 'filename']
# of duplicates: 2528 --> columns dropped: ['author', 'url', 'filename']
# of duplicates: 2505 --> columns dropped: ['license', 'url', 'filename']
# of duplicates: 3485 --> columns dropped: ['rating', 'url', 'filename']
# of duplicates: 2503 --> columns dropped: ['url', 'filename']
# of duplicates: 2503 --> columns dropped: ['filename', 'url']
```

Out[30]:
```
litegr     140
comsan     101
cohmar1     87
eubeat1     86
thrnig1     76
           ...
reedov1      1
refwar2      1
scthon1      1
slbgre1      1
klacuc1      1
Name: primary_label, Length: 201, dtype: int64
```

In [31]:
```
# Function used to find duplicate audio files - inefficient, use audio durations instead
def get_files(df, common_name):
    temp_df = df.query("common_name == @common_name")
    sorted_files = sorted(list(temp_df["filename"]))
    file_list = [os.path.join("/kaggle/input/birdclef-2023/train_audio/", file) for file in sorted_files]
    return file_list
```

In [33]:
```
def visualize_duplicate_audio(file1, file2):

    path = "/kaggle/input/birdclef-2023/train_audio/"
    filename1 = path + file1
    filename2 = path + file2

    fig, axs = plt.subplots(1, 2, figsize=(16, 8))

    y1, sr1 = librosa.load(filename1)
    spec1 = librosa.display.specshow(librosa.power_to_db(librosa.feature.melspectrogram(y=y1, sr=sr1, n_mels=128, fmax=800
0)),
                                     y_axis='mel', fmax=8000, x_axis='time', ax=axs[0])

    axs[0].set_title(f'Mel-Spectrogram | file: {filename1.split("/")[-1]}')
    plt.colorbar(spec1, format='%+2.0f dB', ax=axs[0])

    y2, sr2 = librosa.load(filename2)
    spec2 = librosa.display.specshow(librosa.power_to_db(librosa.feature.melspectrogram(y=y2, sr=sr2, n_mels=128, fmax=800
0)),
                                     y_axis='mel', fmax=8000, x_axis='time', ax=axs[1])

    axs[1].set_title(f'Mel-Spectrogram | file: {filename2.split("/")[-1]}')
    plt.colorbar(spec2, format='%+2.0f dB', ax=axs[1])

    print(" " * 5, filename1.split("audio")[1])
    display(Audio(filename1))
    print(" " * 5, filename2.split("audio")[1])
    display(Audio(filename2))

    plt.show()
```

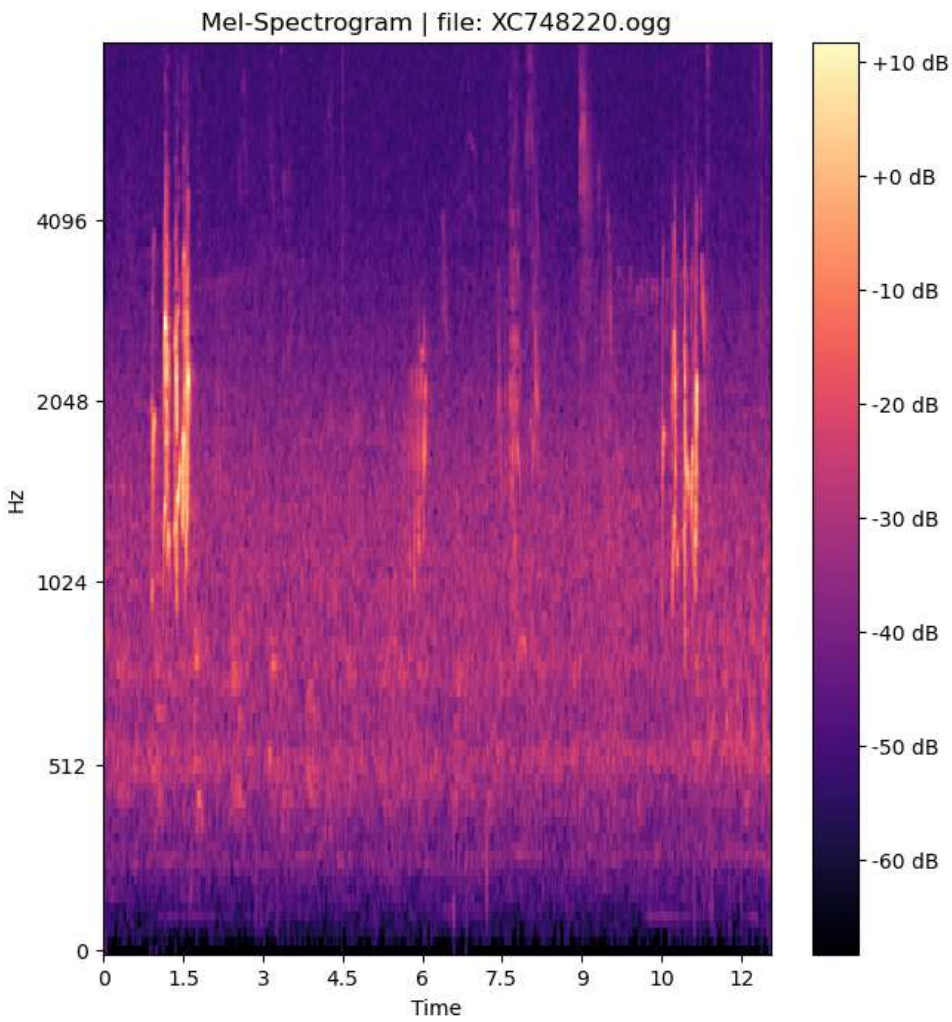
In [34]:
```
visualize_duplicate_audio("combul2/XC748220.ogg", "combul2/XC748221.ogg") # Common Bulbul
```

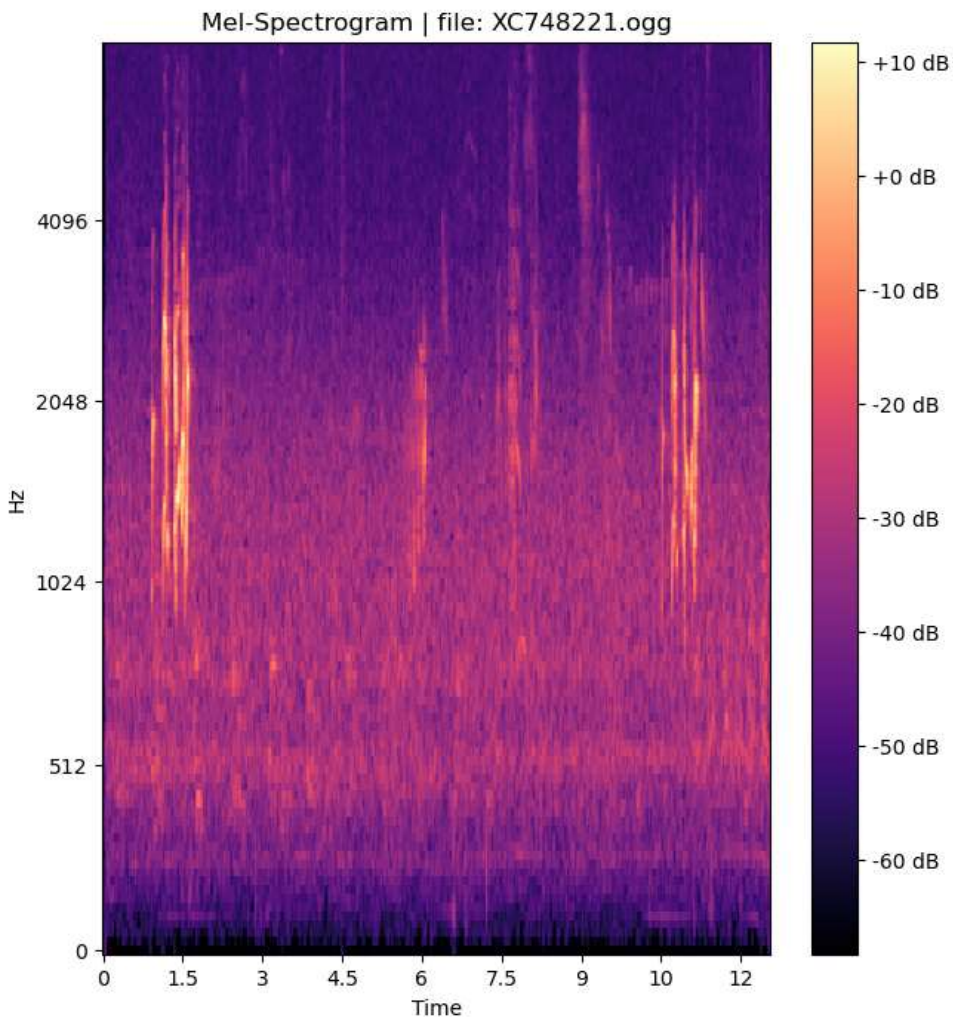
/combul2/XC748220.ogg

0:00 / 0:12

/combul2/XC748221.ogg

0:00 / 0:12

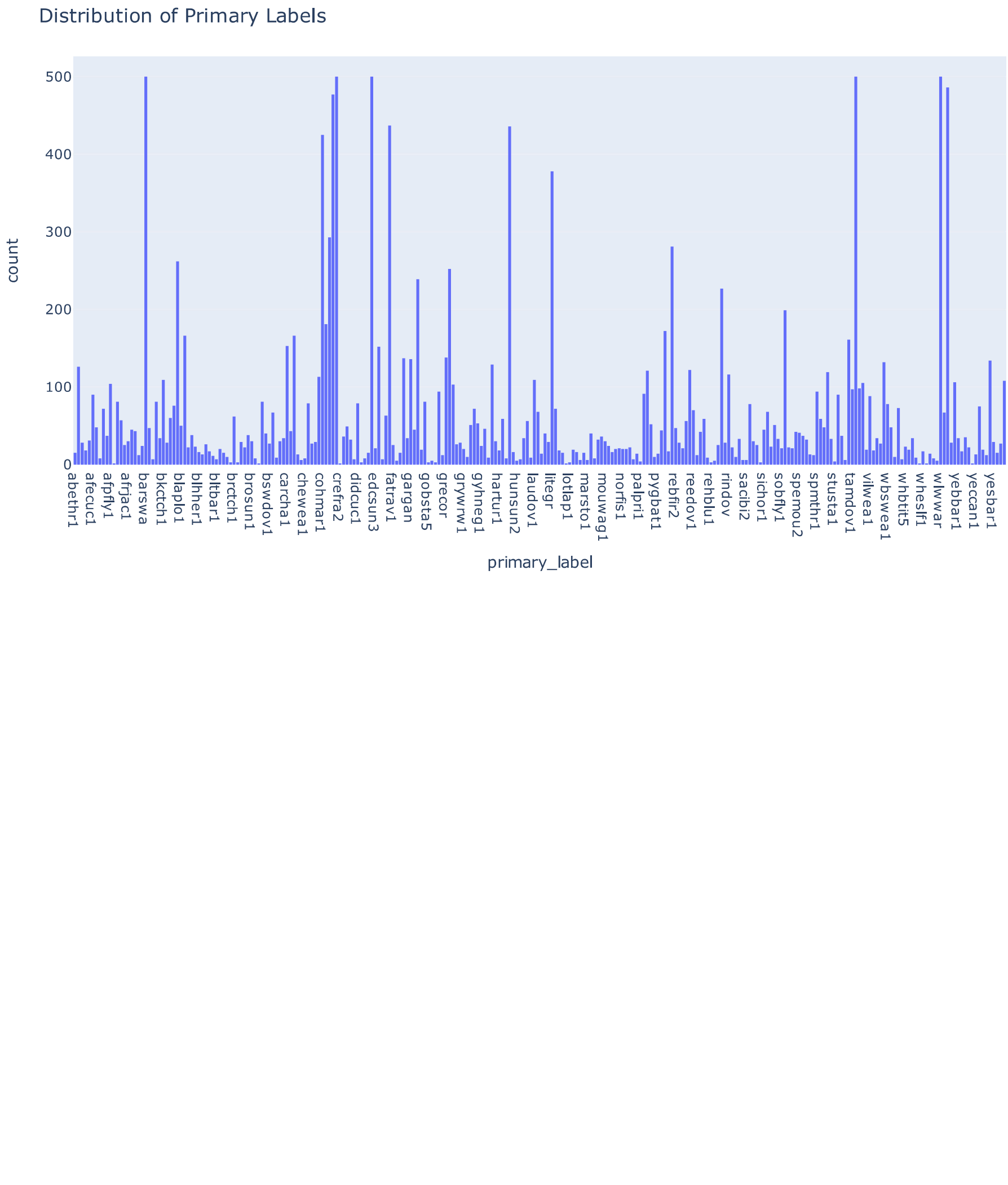




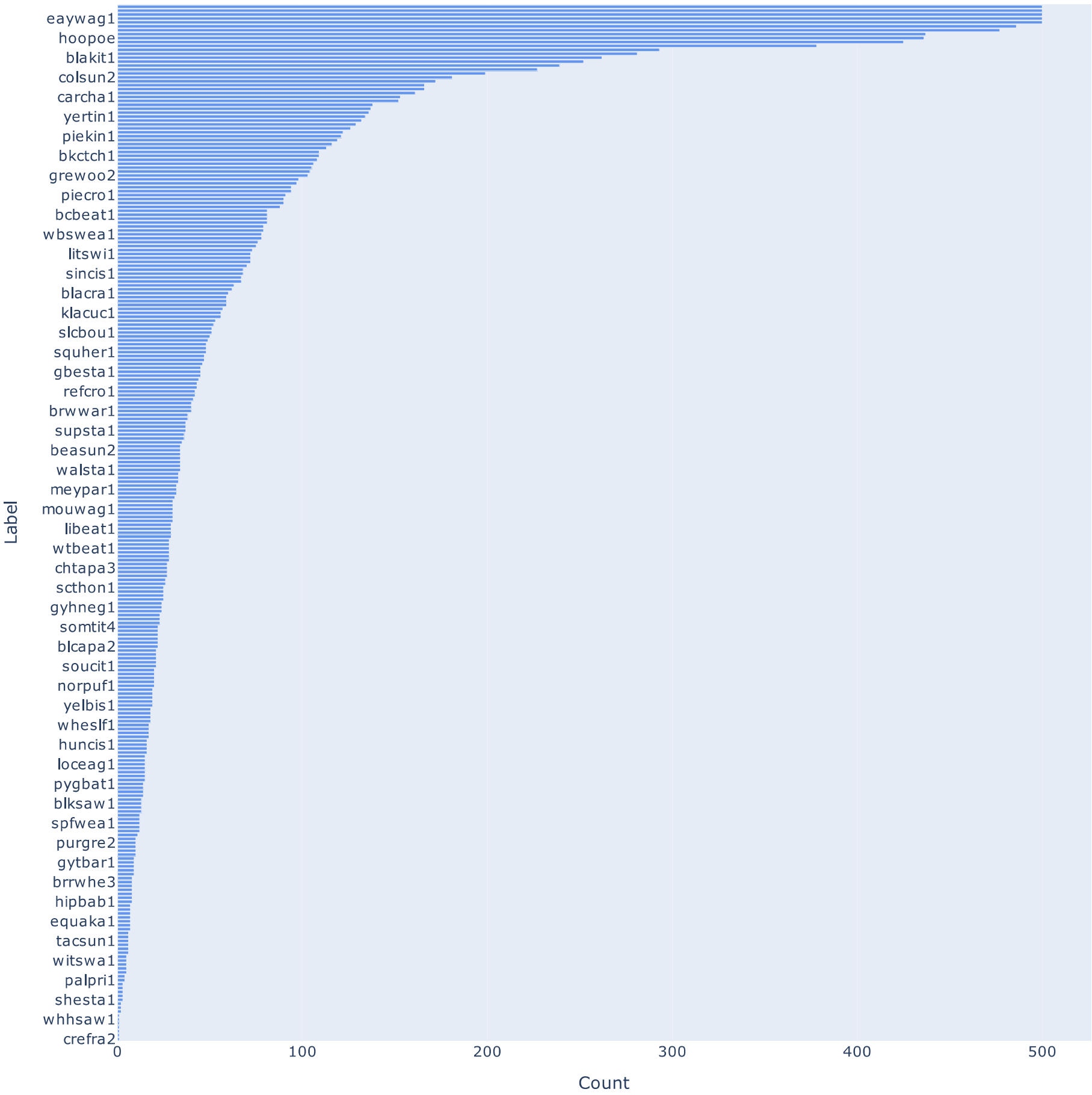
Plots

Top ↑

In [6]:
```
fig = px.histogram(trainmeta_df, x="primary_label", nbins=len(trainmeta_df["primary_label"].unique()))
fig.update_layout(title_text="Distribution of Primary Labels")
fig.show()
```

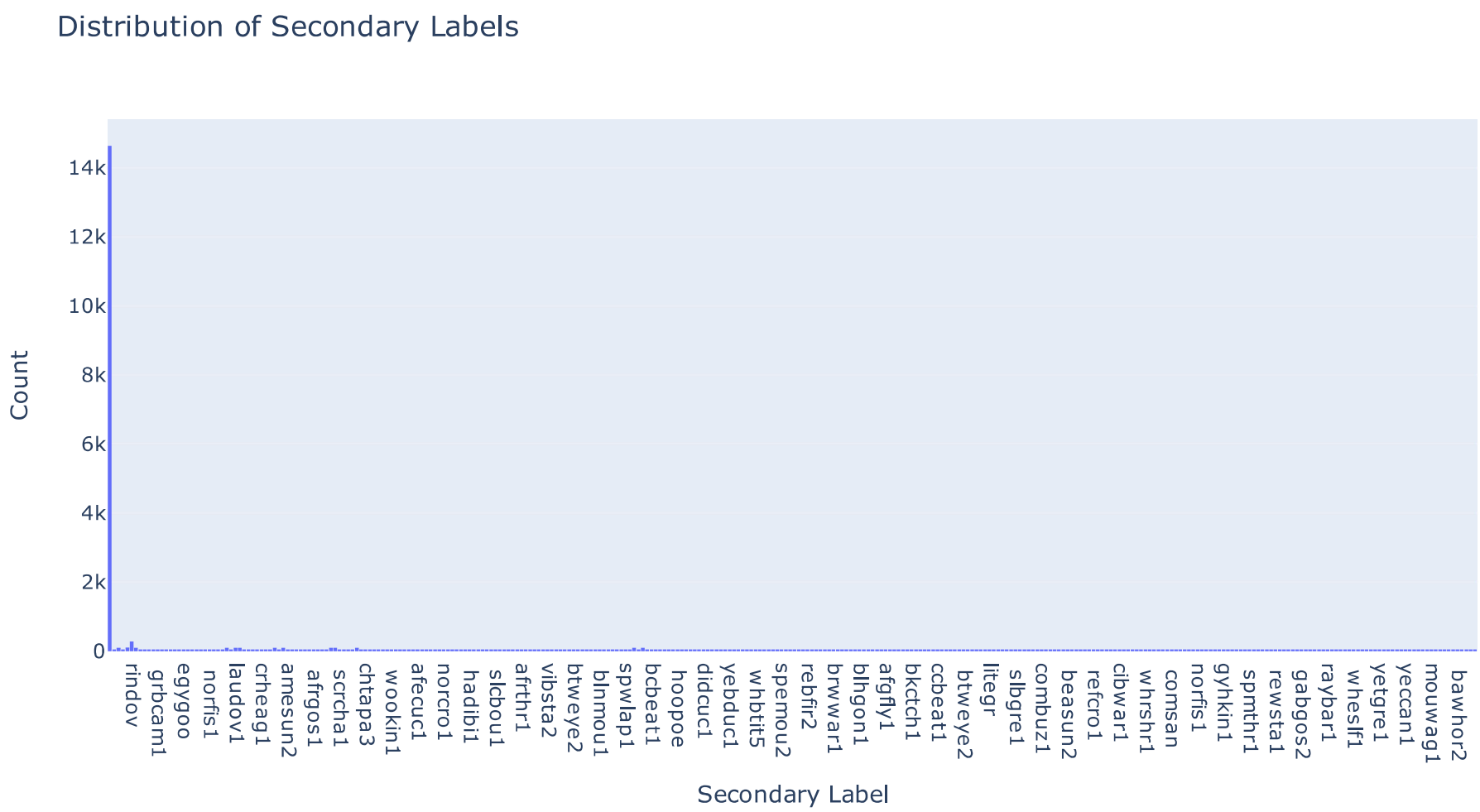


In [22]:
```
primary_label_counts = trainmeta_df.primary_label.value_counts().sort_values()

fig = px.bar(
    x=primary_label_counts.values, y=primary_label_counts.index,
    color_discrete_sequence=['cornflowerblue'],
    orientation='h', height=1000
)
fig.update_layout(xaxis_title="Count", yaxis_title="Label")
fig.show()
```



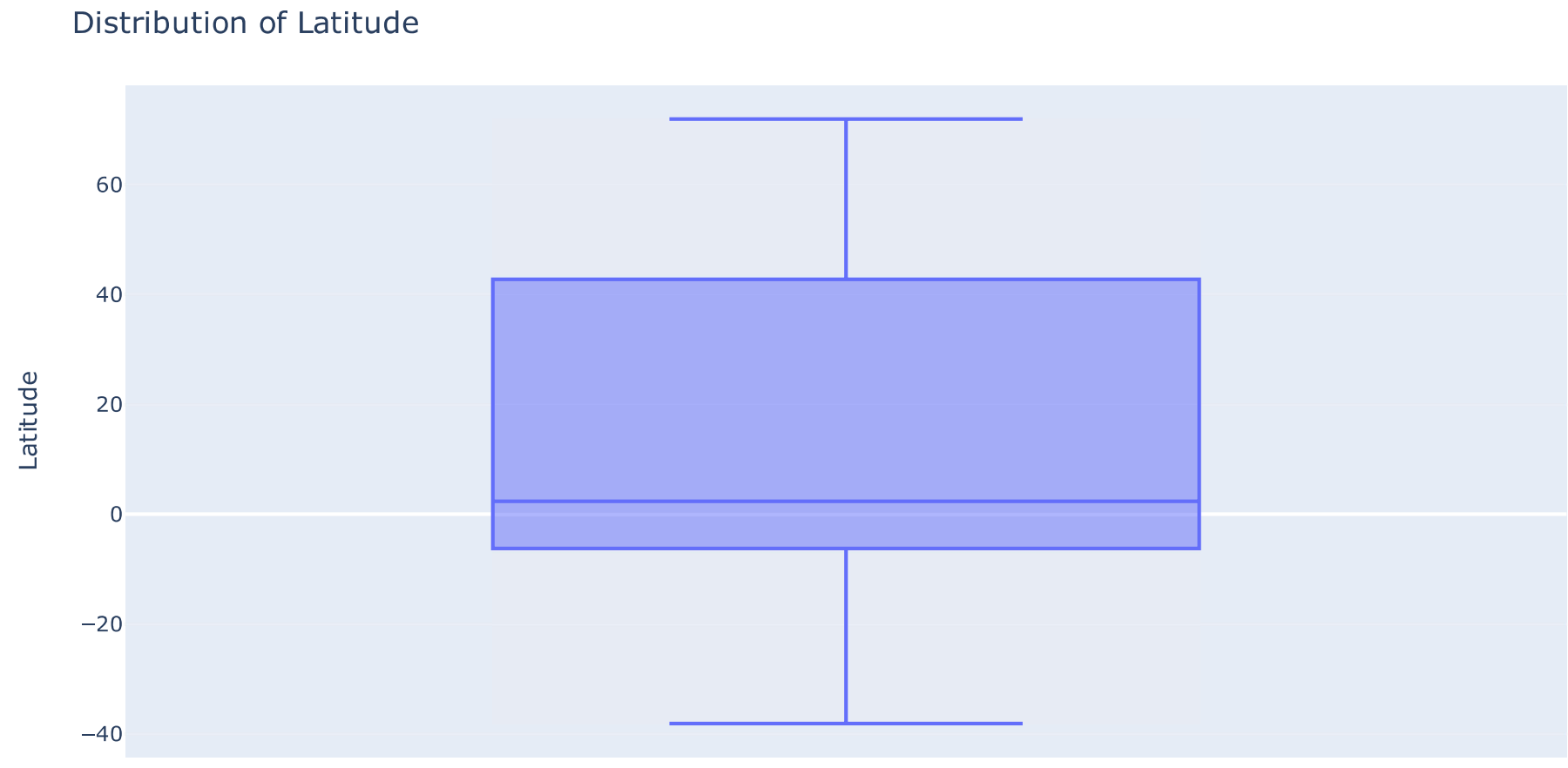
In [7]:
```
secondary_df = trainmeta_df['secondary_labels'].str.replace('[','').str.replace(']','').str.replace('\'','').str.split
(',', expand=True).stack().reset_index(level=1, drop=True).rename('secondary_label')
secondary_df = pd.merge(trainmeta_df.drop(columns=['secondary_labels']), secondary_df, left_index=True, right_index=True)

fig = px.histogram(secondary_df, x="secondary_label", title="Distribution of Secondary Labels")
fig.update_layout(xaxis_title="Secondary Label", yaxis_title="Count")
fig.show()
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning:

The default value of regex will change from True to False in a future version. In addition, single character regular expres
sions will *not* be treated as literal strings when regex=True.
```

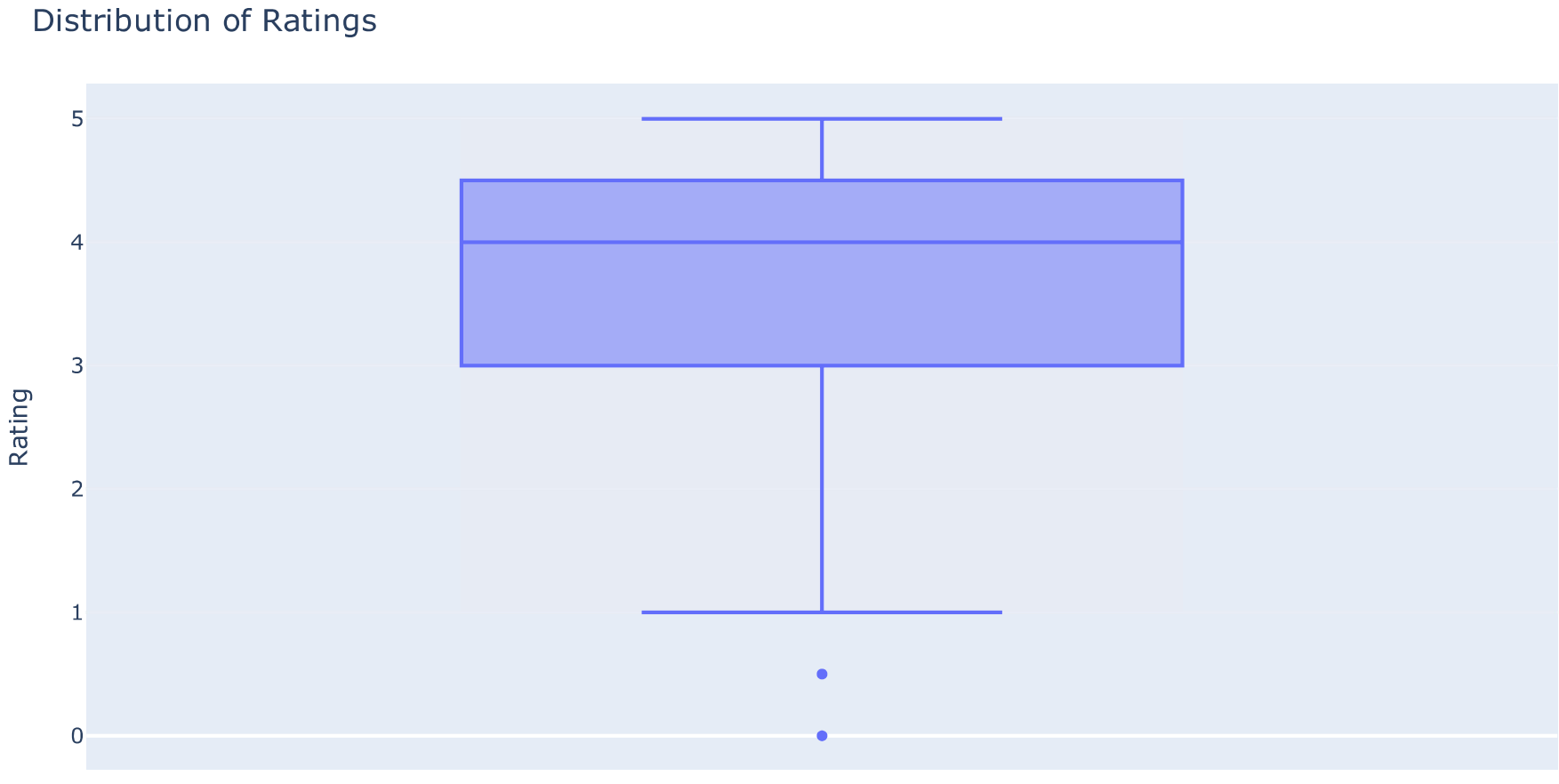


In [8]:
```
# Flatten the list of labels in the "type" column
labels = [label.strip("[]'") for sublist in trainmeta_df['type'].apply(ast.literal_eval) for label in sublist]

# Count the occurrence of each label
label_counts = Counter(labels)

# Create a bar plot of the label counts
fig = px.bar(x=list(label_counts.keys()), y=list(label_counts.values()))
fig.update_layout(title_text="Distribution of Types")
fig.show()
```

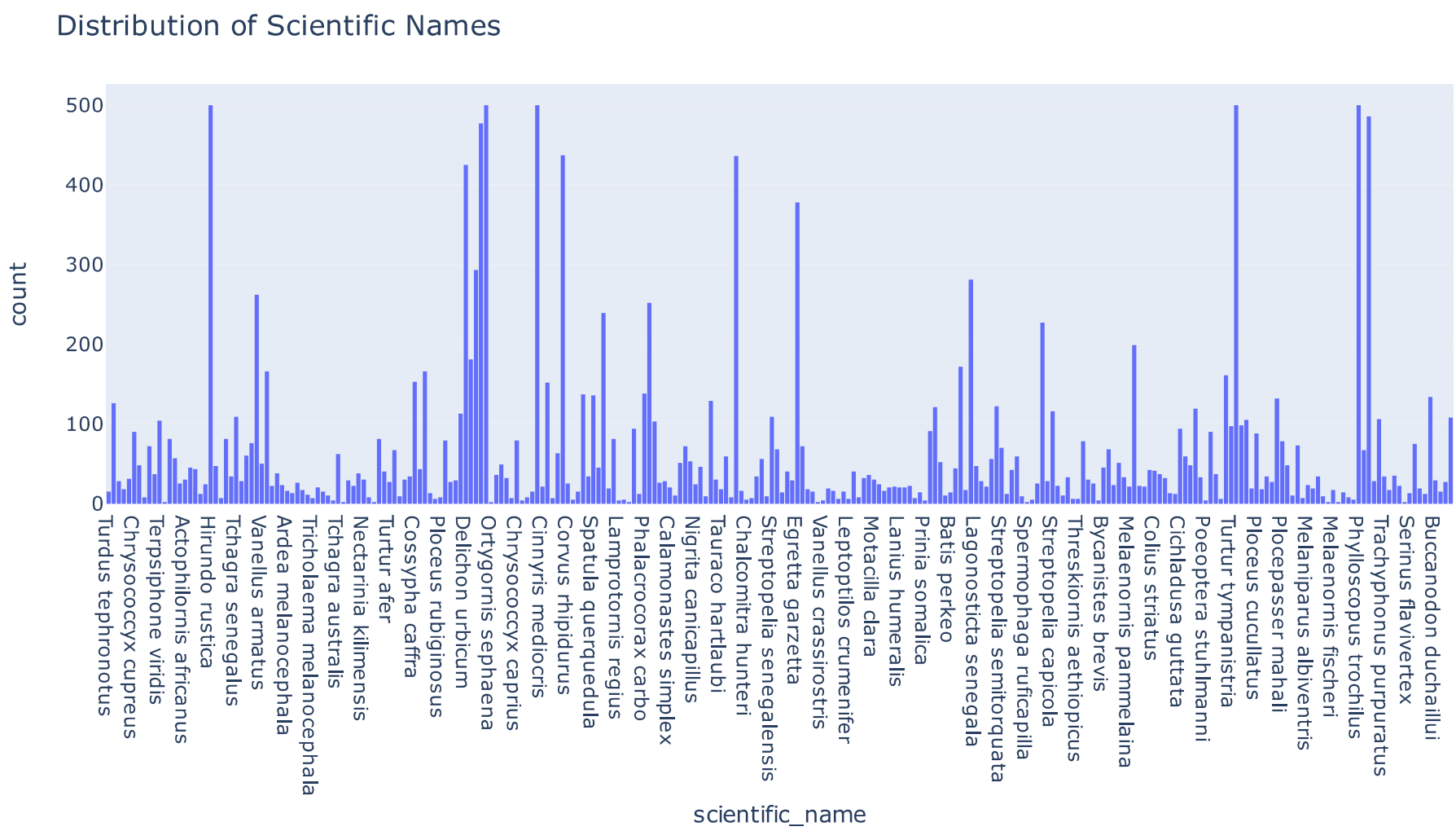
In [9]:
```
fig1 = px.box(trainmeta_df, y="latitude")
fig1.update_layout(title_text="Distribution of Latitude",
                   yaxis=dict(title="Latitude"))

# create a box plot for longitude
fig2 = px.box(trainmeta_df, y="longitude")
fig2.update_layout(title_text="Distribution of Longitude",
                   yaxis=dict(title="Longitude"))

# create a box plot for rating
fig3 = px.box(trainmeta_df, y="rating")
fig3.update_layout(title_text="Distribution of Ratings",
                   yaxis=dict(title="Rating"))

# show the figures
fig1.show()
fig2.show()
fig3.show()
```


Distribution of Latitude
Latitude
60
40
20
0
−20
−40

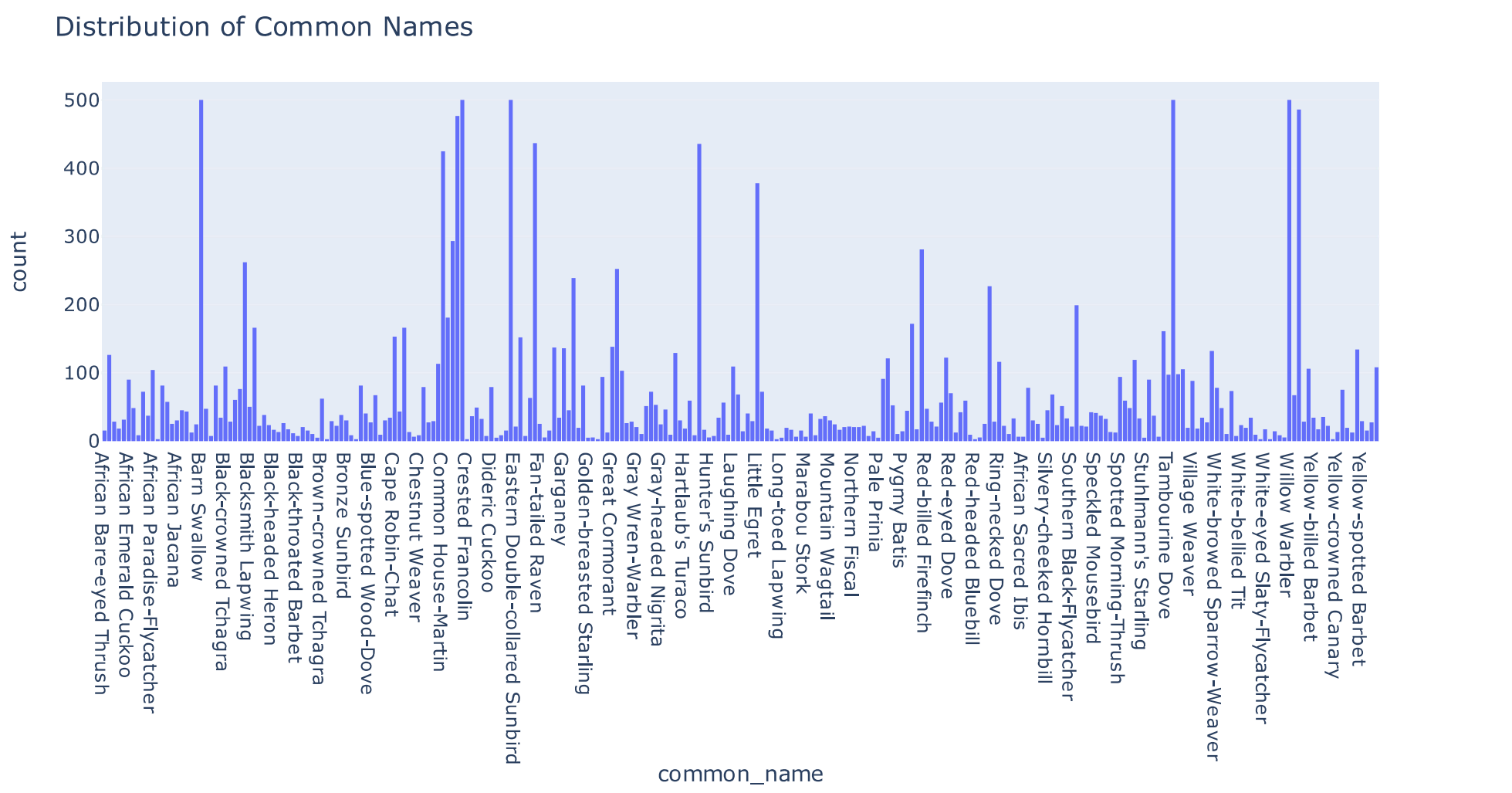
Distribution of Longitude
Longitude
150
100
50
0
−50
−100
−150



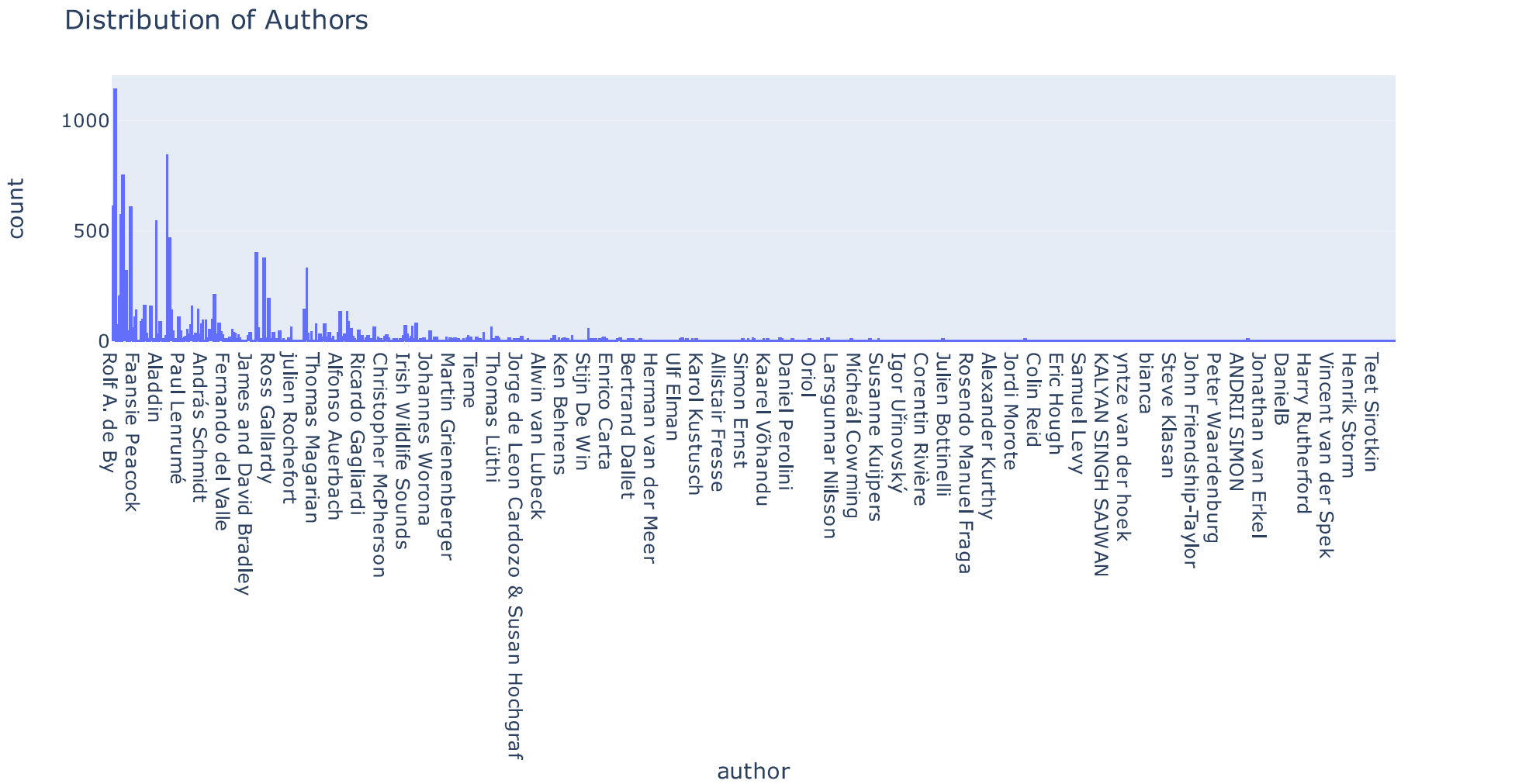
In [10]:
```
fig = px.histogram(trainmeta_df, x="scientific_name", nbins=len(trainmeta_df["scientific_name"].unique()))
fig.update_layout(title_text="Distribution of Scientific Names")
fig.show()
```



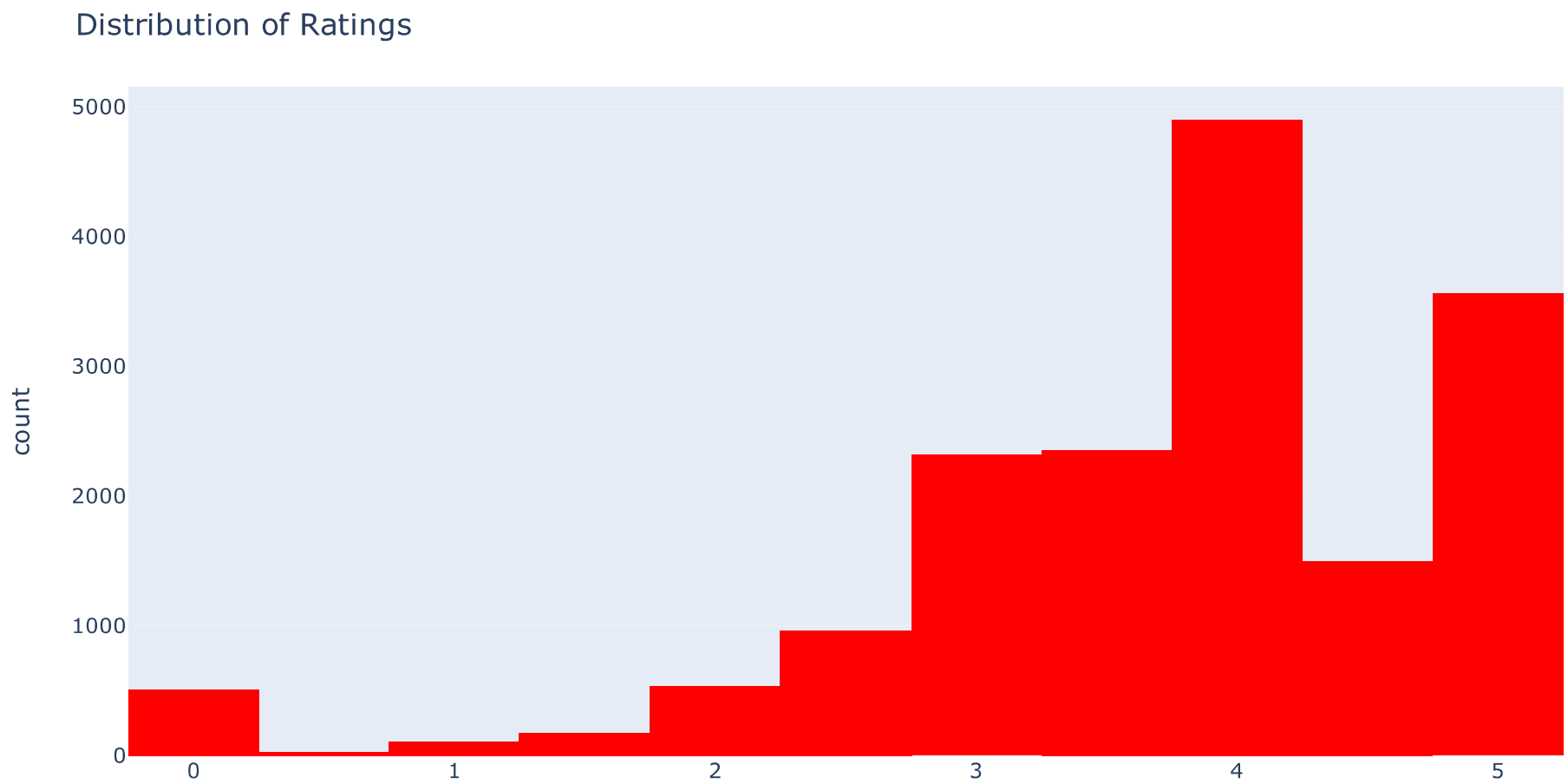
In [11]:
```
fig = px.histogram(trainmeta_df, x="common_name", nbins=len(trainmeta_df["common_name"].unique()))
fig.update_layout(title_text="Distribution of Common Names")
fig.show()
```



In [12]:
```
fig = px.histogram(trainmeta_df, x="author", nbins=len(trainmeta_df["author"].unique()))
fig.update_layout(title_text="Distribution of Authors")
fig.show()
```

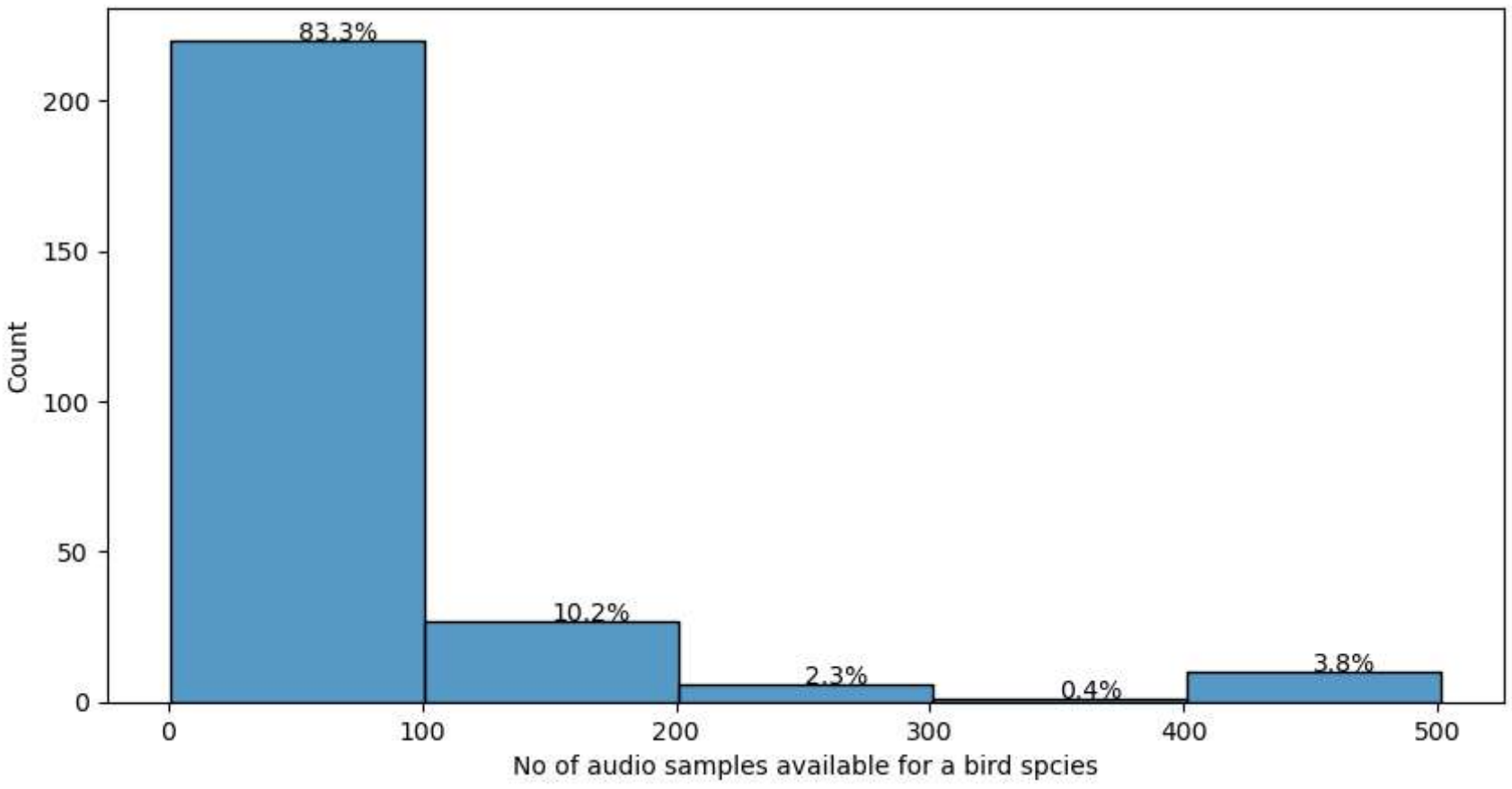


In [13]:
```
fig = px.histogram(trainmeta_df, x="rating", nbins=len(trainmeta_df["rating"].unique()) , color_discrete_sequence=['red'])
fig.update_layout(title_text="Distribution of Ratings")
fig.show()
```



Audio Samples

In [37]:
```
# binning the dataset based on number of audio samples avaiable
plt.figure(figsize=(10, 5)) # width, height (inches)

temp = trainmeta_df['primary_label'].value_counts()
ax = sns.histplot(temp, binwidth=100)
totals = []
for p in ax.patches:
    totals.append(p.get_height())
total = sum(totals)
for p in ax.patches:
    percentage = '{:.1f}%'.format(100 * p.get_height()/total)
    x = p.get_x() + (p.get_width() / 2)
    y = p.get_y() + p.get_height() + 0.01
    ax.annotate(percentage, (x, y))

plt.xlabel('No of audio samples available for a bird spcies')
plt.show()
```



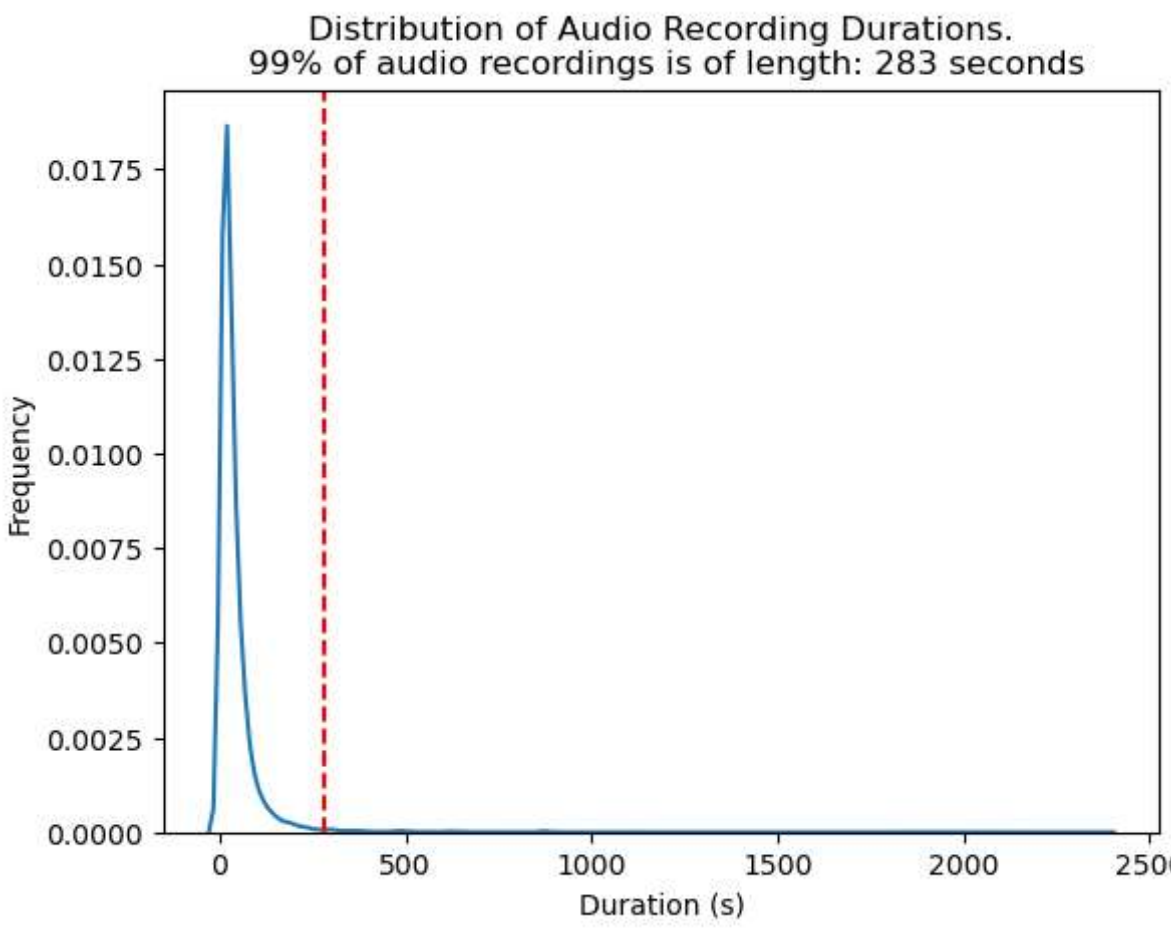
Audio Duration

In [44]:
```
import multiprocessing
from tqdm.auto import tqdm
```

In [40]:
```
audio_path = '/kaggle/input/birdclef-2023/train_audio'
```

In [47]:
```
#ref: https://stackoverflow.com/a/45276885/12828621

#NOTE: without mutliprocessing this cells takes more than 1 hour to run,
#but with mutltiprocesing it runs in less than 5 minutes

# plot the length of audio recordings as a distribution
# list of audio file paths
trainmeta_df['path'] = audio_path + '/' + trainmeta_df['filename']
audio_files = trainmeta_df['path'].tolist()

# get the duration of each file in seconds (using multiprocessing) (we have in kaggle 4cores)
def calculate_audio_length(filepath):
    y, sr = librosa.load(filepath, sr=None)
    length = librosa.get_duration(y=y, sr=sr)
    return length

n_cores = multiprocessing.cpu_count()
print('Available cores: ', n_cores)
with multiprocessing.Pool(n_cores) as p:
    duration = list(tqdm(p.imap(calculate_audio_length, audio_files), total=len(audio_files)))


#plot
sns.kdeplot(x=duration)
percentile = np.percentile(duration, 99)
print('99 % of audio recordings is of length: ', round(percentile), ' seconds')
plt.axvline(x=percentile, color='r', linestyle='--')
plt.xlabel("Duration (s)")
plt.ylabel("Frequency")
plt.title(f"Distribution of Audio Recording Durations.\n 99% of audio recordings is of length: {round(percentile)} second
s")
plt.show()
```

```
Available cores:  4

99 % of audio recordings is of length:  283  seconds
```

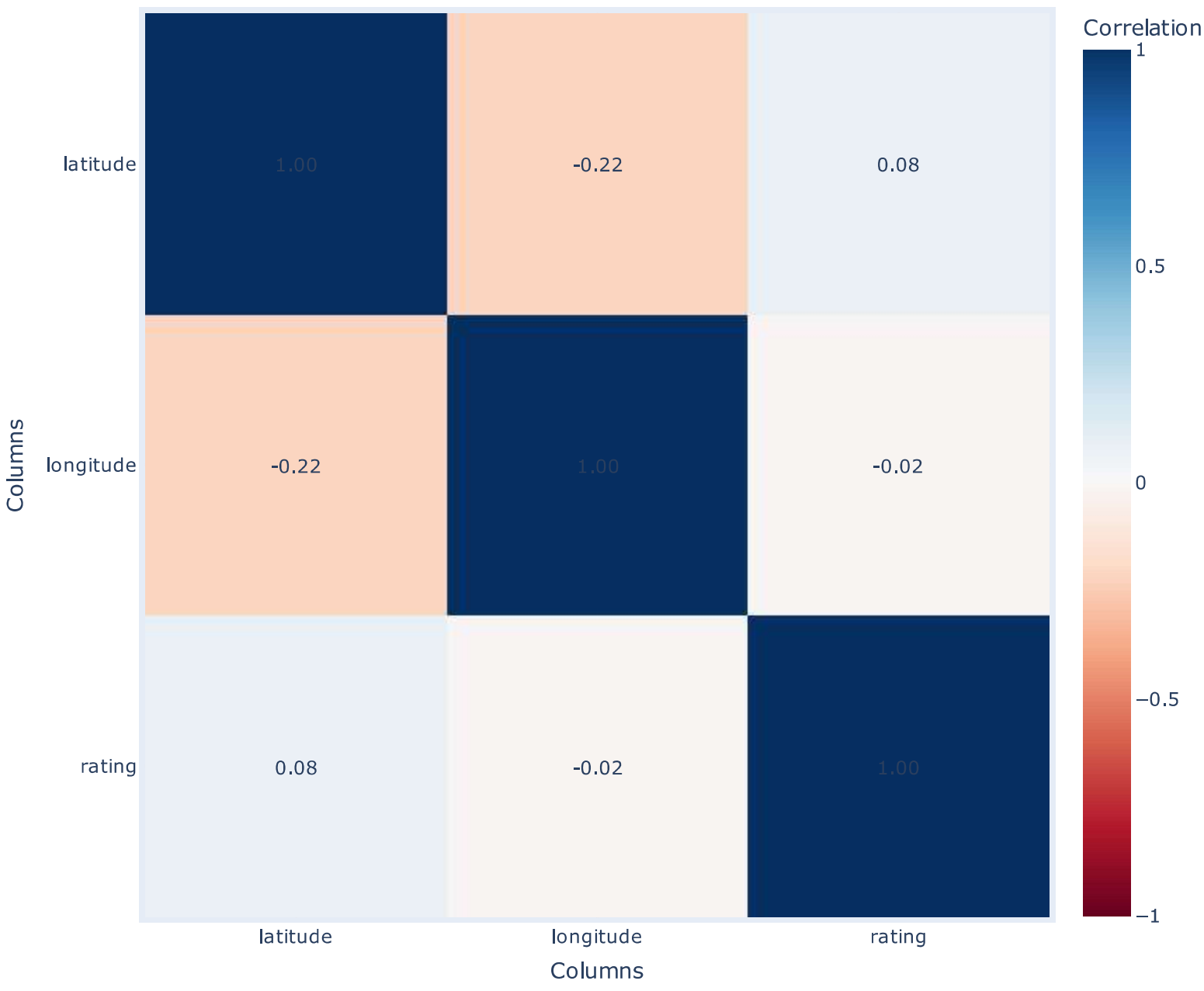



```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_27/1880538918.py in <module>
     29 plt.ylabel("Frequency")
     30 plt.title(f"Distribution of Audio Recording Durations.\n 99% of audio recordings is of length: {round(percentile)}
seconds")
---> 31 plt.show()['path'].tolist()
     32
     33 # get the duration of each file in seconds (using multiprocessing) (we have in kaggle 4cores)

TypeError: 'NoneType' object is not subscriptable
```

Correlation Plots

Top ↑

In [14]:
```
# drop columns from correlation matrix
corr = trainmeta_df.corr()

# create correlation heatmap
fig = px.imshow(corr,
                labels=dict(x="Columns", y="Columns", color="Correlation"),
                x=corr.columns,
                y=corr.columns,
                color_continuous_scale='RdBU',
                zmin=-1,
                zmax=1,
                title="Correlation Heatmap")

# add text annotations
annotations = []
for i, row in enumerate(corr.values):
    for j, value in enumerate(row):
        text = '{:.2f}'.format(value)
        annotations.append(dict(x=corr.columns[j], y=corr.columns[i], text=text, showarrow=False))

fig.update_layout(width=800, height=800)
fig.update_traces(showscale=True, colorbar_thickness=25, colorbar_len=0.75)
fig.update_layout(margin=dict(l=50, r=50, b=100, t=100, pad=4))
fig.update_layout(annotations=annotations)
fig.show()
```



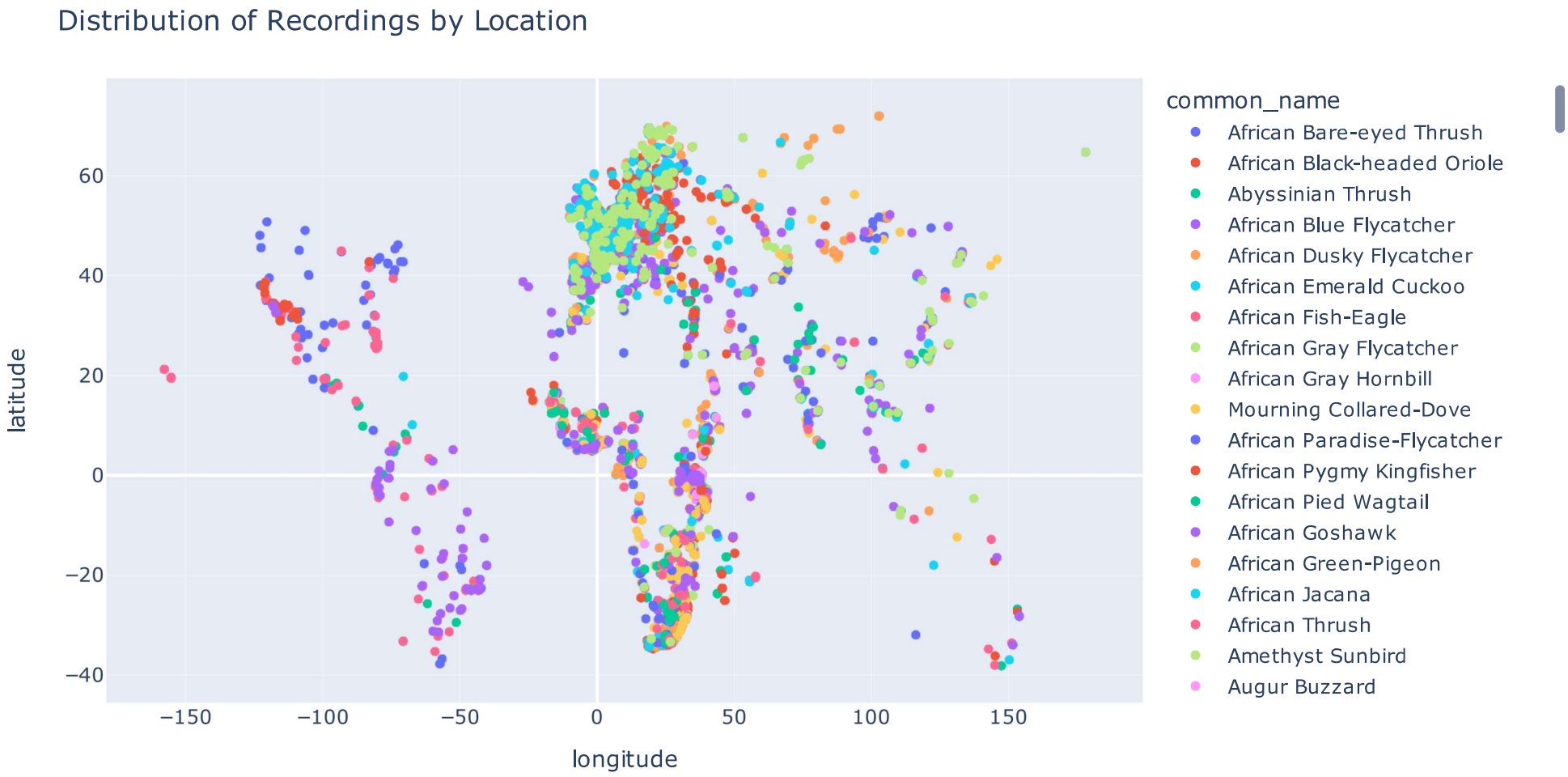
Interactive Map Plots

Top ↑

Scatter Plot

Top ↑

In [15]:
```
fig = px.scatter(trainmeta_df, x="longitude", y="latitude", color="common_name")
fig.update_layout(title="Distribution of Recordings by Location")
fig.show()
```



Map Box (Open Street Map)

Top ↑

In [48]:
```
# fig = px.scatter_mapbox(trainmeta_df, lat="latitude", lon="longitude", color="common_name",
#                         hover_name="filename", hover_data=["common_name", "author", "rating"],
#                         zoom=3, height=500)
# fig.update_layout(mapbox_style="open-street-map")
# fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
# fig.show()
```

Map Box(Terrain View)

Top ↑

In [49]:
```
# fig = px.density_mapbox(trainmeta_df, lat='latitude', lon='longitude', radius=10,
#                         center=dict(lat=47.6, lon=-122.3), zoom=7,
#                         mapbox_style="stamen-terrain")
# fig.update_layout(title_text="Distribution of Bird Sightings")
# fig.show()
```

EBird Taxonomy

Top ↑

In [18]:
```
train_species_df = pd.read_csv("/kaggle/input/birdclef-2023/eBird_Taxonomy_v2021.csv")
train_species_df.head()
```

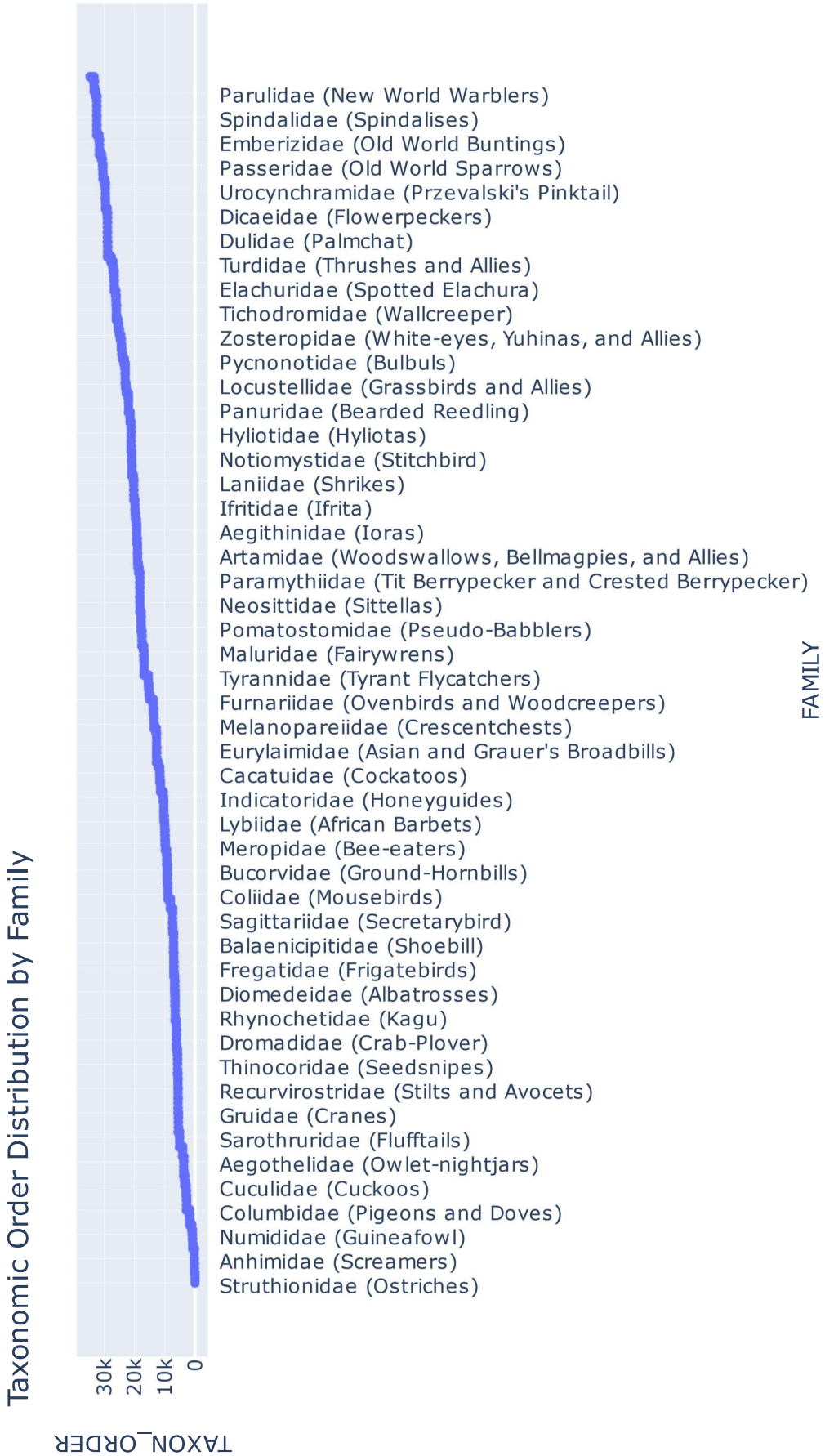
Out[18]:

	TAXON_ORDER	CATEGORY	SPECIES_CODE	PRIMARY_COM_NAME	SCI_NAME	ORDER1	FAMILY	SPECIES_GROUP	REPORT_AS
0	1	species	ostric2	Common Ostrich	Struthio camelus	Struthioniformes	Struthionidae (Ostriches)	Ostriches	NaN
1	6	species	ostric3	Somali Ostrich	Struthio molybdophanes	Struthioniformes	Struthionidae (Ostriches)	NaN	NaN
2	7	slash	y00934	Common/Somali Ostrich	Struthio camelus/molybdophanes	Struthioniformes	Struthionidae (Ostriches)	NaN	NaN
3	8	species	grerhe1	Greater Rhea	Rhea americana	Rheiformes	Rheidae (Rheas)	Rheas	NaN
4	14	species	lesrhe2	Lesser Rhea	Rhea pennata	Rheiformes	Rheidae (Rheas)	NaN	NaN

In [19]:
```
# Histogram of the taxonomic order counts
fig1 = px.histogram(train_species_df, x="TAXON_ORDER", color_discrete_sequence=['aquamarine'])
fig1.update_layout(title_text="Distribution of Taxonomic Orders")

# Bar plot of the species group counts
# Box plot of the taxonomic order counts by category
fig3 = px.box(train_species_df, x="CATEGORY", y="TAXON_ORDER", color_discrete_sequence=['red'])
fig3.update_layout(title_text="Taxonomic Order Distribution by Category")

# Scatter plot of the taxonomic order counts by family
fig4 = px.scatter(train_species_df, x="FAMILY", y="TAXON_ORDER")
fig4.update_layout(title_text="Taxonomic Order Distribution by Family")

# Show the plots
fig1.show()
fig3.show()
fig4.show()
```


Distribution of Taxonomic Orders
count
0
50
100
150
200
250
300
350
0
5k
10k
15k
20k
25k
30k
35k
TAXON_ORDER

Taxonomic Order Distribution by Category
TAXON_ORDER
0
5k
10k
15k
20k
25k
30k
35k
species
slash
issf
spuh
hybrid
domestic
form
intergrade
CATEGORY



3. AUDIO EXPLORATION

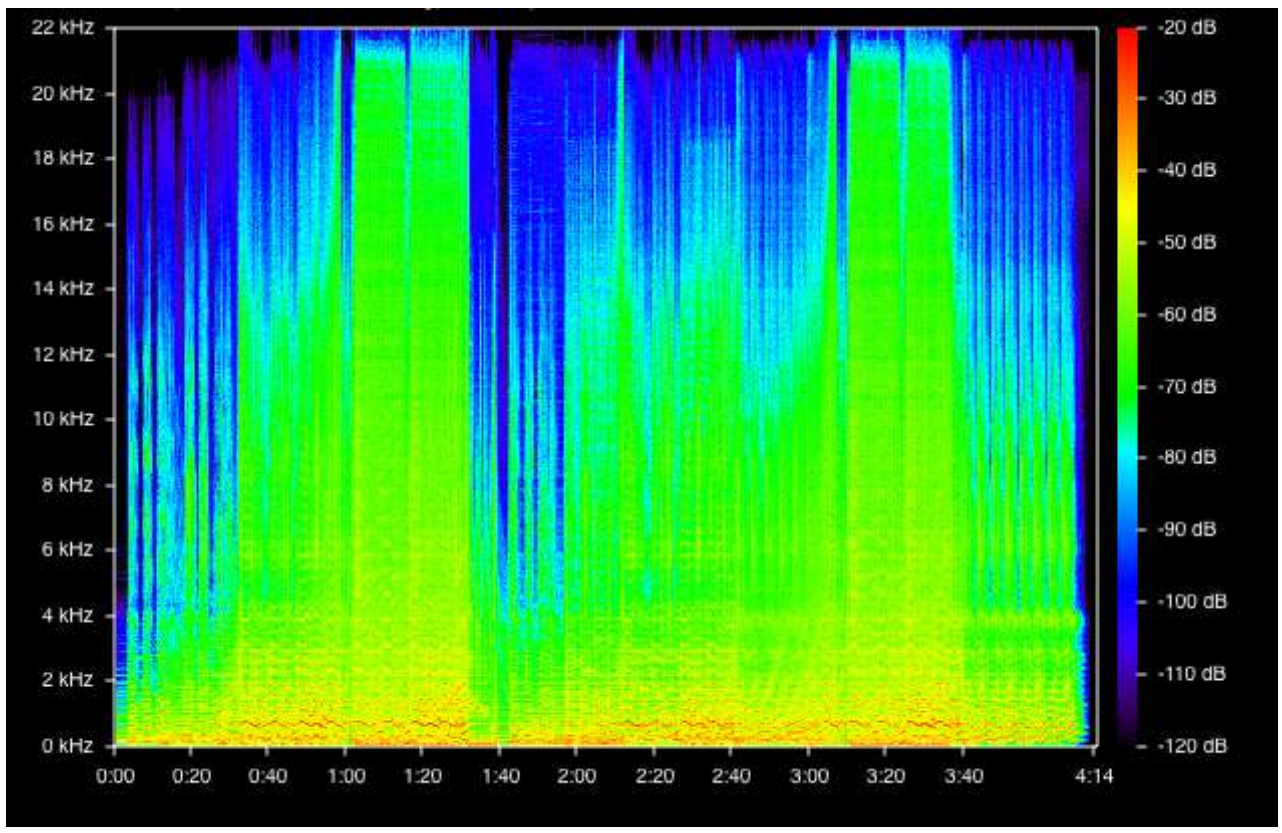
Top ↑

How to Visualize Audio Files ??

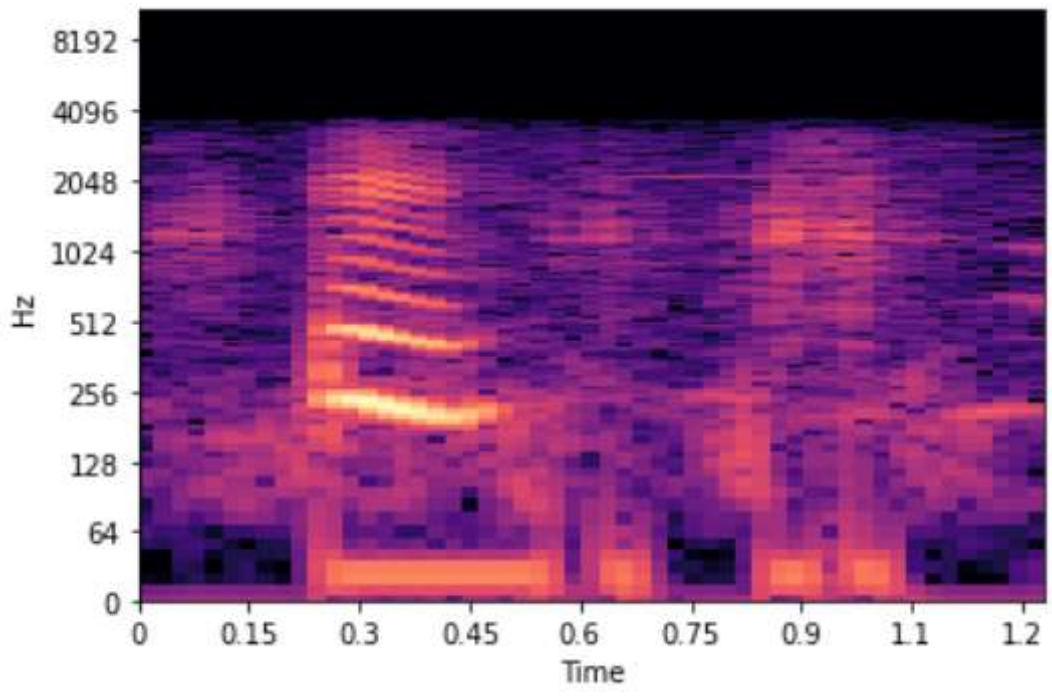
There are many ways in which we can view audio in 2D like :

1. **Waveforms :** In audio processing, a waveform is a graphical representation of a sound signal that shows how the signal varies over time. It is a plot of the amplitude of the sound wave on the y-axis versus time on the x-axis. Waveforms can be used to visualize and analyze the properties of audio signals, such as frequency, amplitude, phase, and duration.

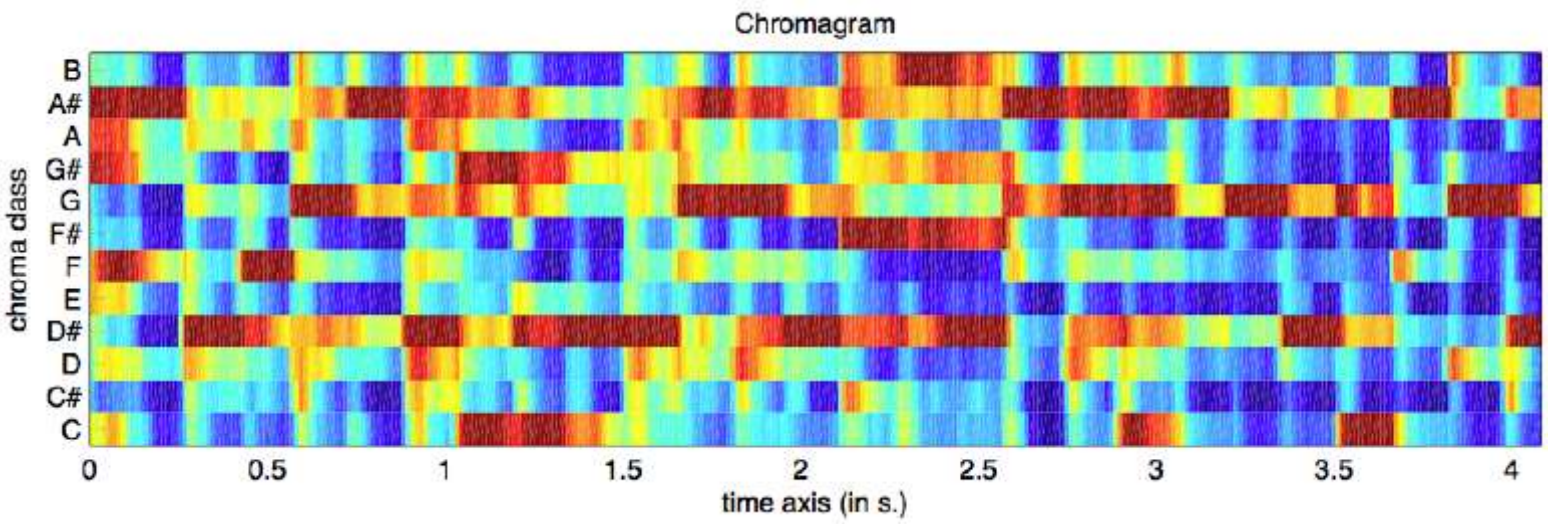
2. **Spectogram :** A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. Spectrum refers to plot/distribution of energy or amplitude as a function of frequency. Spectrogram is commonly used in signal processing, audio analysis, and other fields to analyze the “frequency content” of a time-varying signal. To create a spectrogram, the signal is first divided into short segments (often using a technique called windowing). Then, for each segment, a mathematical transformation, such as the Fast Fourier Transform (FFT), is applied to convert the signal from the time domain to the frequency domain. The result is a set of frequency components and their respective amplitudes for that particular time segment.



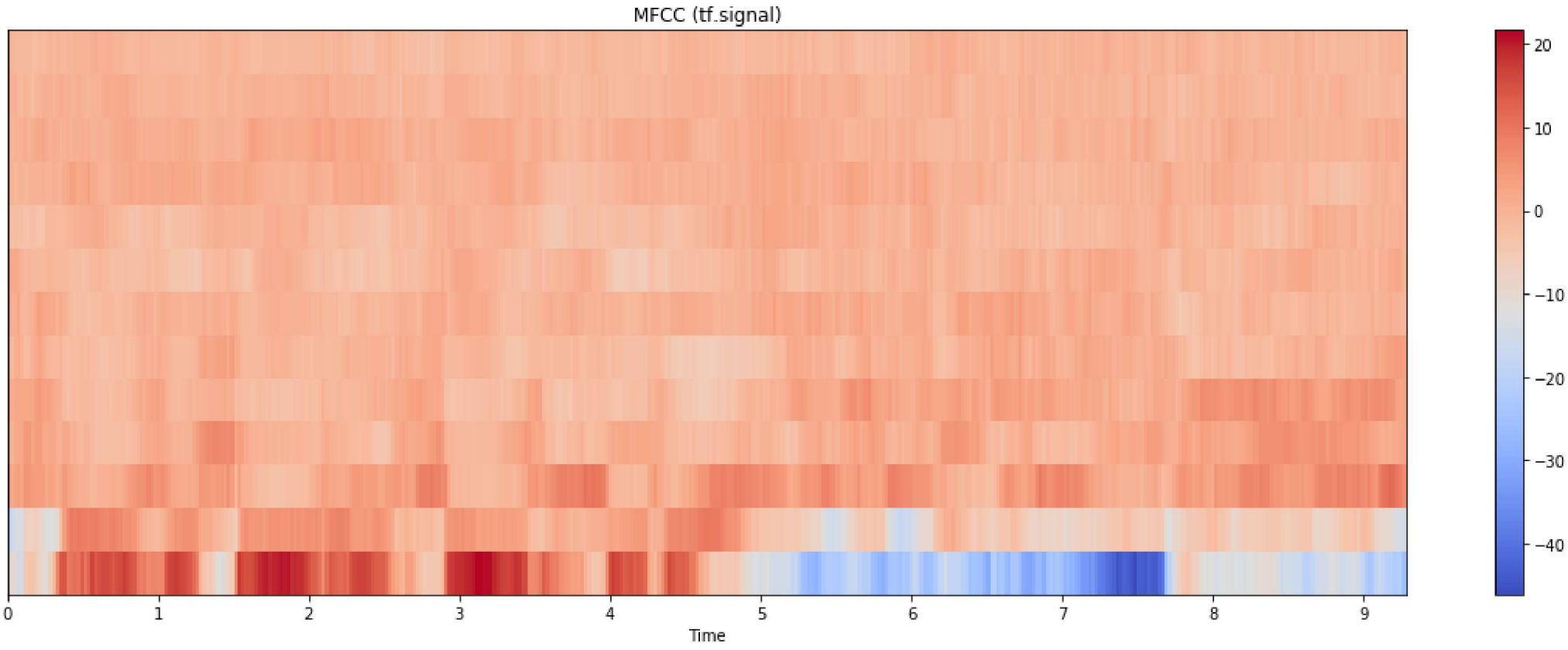
3. **Mel Spectograms :** The FFT (fast fourier transform) is computed on overlapping windowed segments of the signal, and we get what is called the spectrogram. A mel spectrogram is a spectrogram where the frequencies are converted to the mel scale. The Mel Scale is a logarithmic transformation of a signal's frequency. The core idea of this transformation is that sounds of equal distance on the Mel Scale are perceived to be of equal distance to humans.



4. **Chromagram :** The term chroma feature or chromagram closely relates to twelve different pitch classes. Chroma-based features, which are also referred to as "pitch class profiles", are a powerful tool for analyzing music whose pitches can be meaningfully categorized (often into twelve categories). A chromagram, is derived from the standard spectrogram, but instead of showing the energy distribution in frequency bins, it displays the energy distribution of musical pitch classes.



5. **MFCC :** Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC (mel-frequency cepstrum). They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale. The mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a “linear cosine transform” of a “log power spectrum” on a “nonlinear mel scale” of frequency. Further, cepstrum is derived from the spectrum of a signal by taking the inverse Fourier transform of the logarithm of the spectrum. It is essentially the spectrum of the spectrum. The term "cepstrum" comes from reversing the letters of the term "spectrum".



Helper Function

Top ↑

In [20]:
```
def audio_eda(audio_path):

    # Load an audio file
    samples, sample_rate = librosa.load(audio_path)

    # Visualize the waveform
    plt.figure(figsize=(14, 5))
    librosa.display.waveshow(samples, sr=sample_rate)
    plt.title('Waveform')

    # Compute the spectrogram
    spectrogram = librosa.stft(samples)
    spectrogram_db = librosa.amplitude_to_db(abs(spectrogram))

    # Visualize the spectrogram
    plt.figure(figsize=(14, 5))
    librosa.display.specshow(spectrogram_db, sr=sample_rate, x_axis='time', y_axis='log')
    plt.colorbar(format='%+2.0f dB')
    plt.title('Spectrogram (dB)')

    # Compute the mel spectrogram


    # Visualize the mel spectrogram
    S = librosa.feature.melspectrogram(y=samples, sr=sample_rate)

    # Visualize mel spectrogram
    plt.figure(figsize=(10, 4))
    librosa.display.specshow(librosa.power_to_db(S, ref=np.max), y_axis='mel', fmax=8000, x_axis='time')
    plt.colorbar(format='%+2.0f dB')
    plt.title('Mel spectrogram')
    plt.tight_layout()


    # Compute the chromagram
    chromagram = librosa.feature.chroma_stft( y = samples , sr = sample_rate)

    # Visualize the chromagram
    plt.figure(figsize=(14, 5))
    librosa.display.specshow(chromagram, sr=sample_rate, x_axis='time', y_axis='chroma')
    plt.colorbar()
    plt.title('Chromagram')

    # Compute the MFCCs
    mfccs = librosa.feature.mfcc(y=samples, sr=sample_rate, n_mfcc=13)

    # Visualize the MFCCs
    plt.figure(figsize=(14, 5))
    librosa.display.specshow(mfccs, sr=sample_rate, x_axis='time')
    plt.colorbar()
    plt.title('MFCCs')

    # Show the plots
    display(Audio(samples, rate=sample_rate))
    plt.show()
```

Audio Exploration

Top ↑

Black-and-white Mannikin

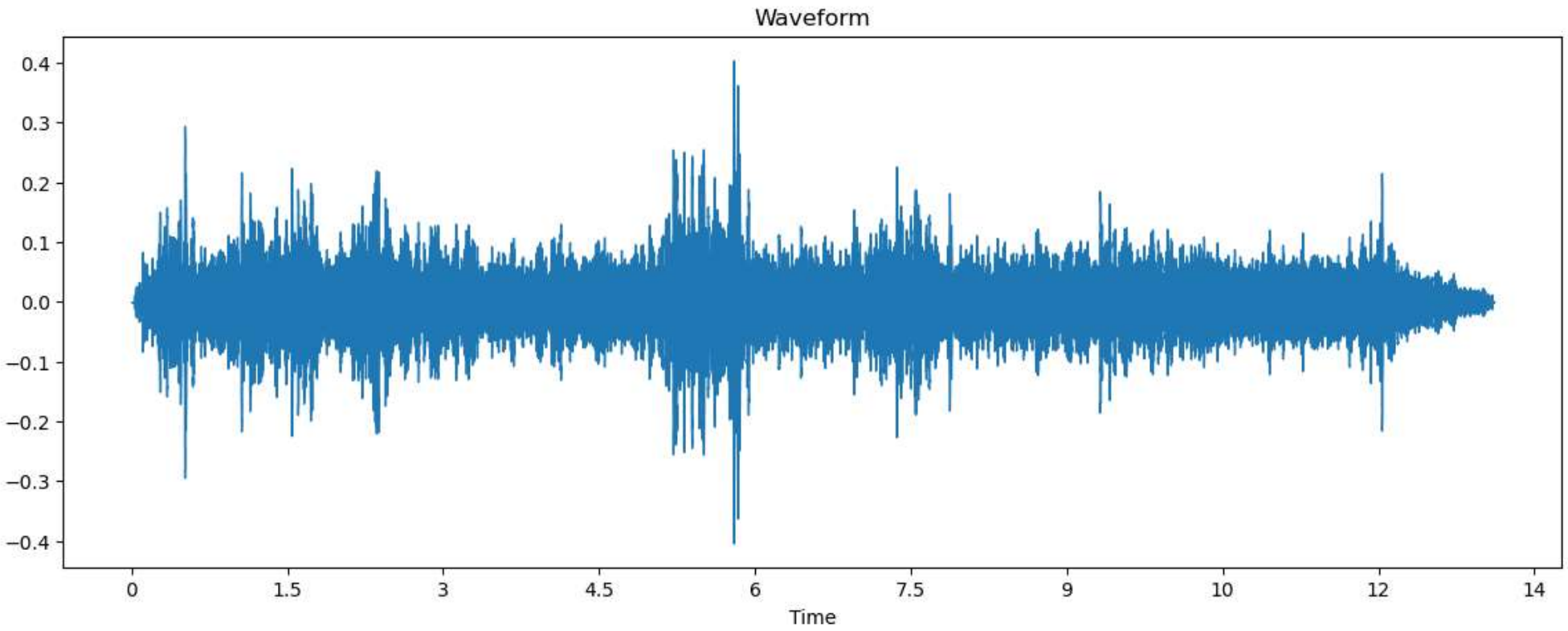
Primary Label : bawman1

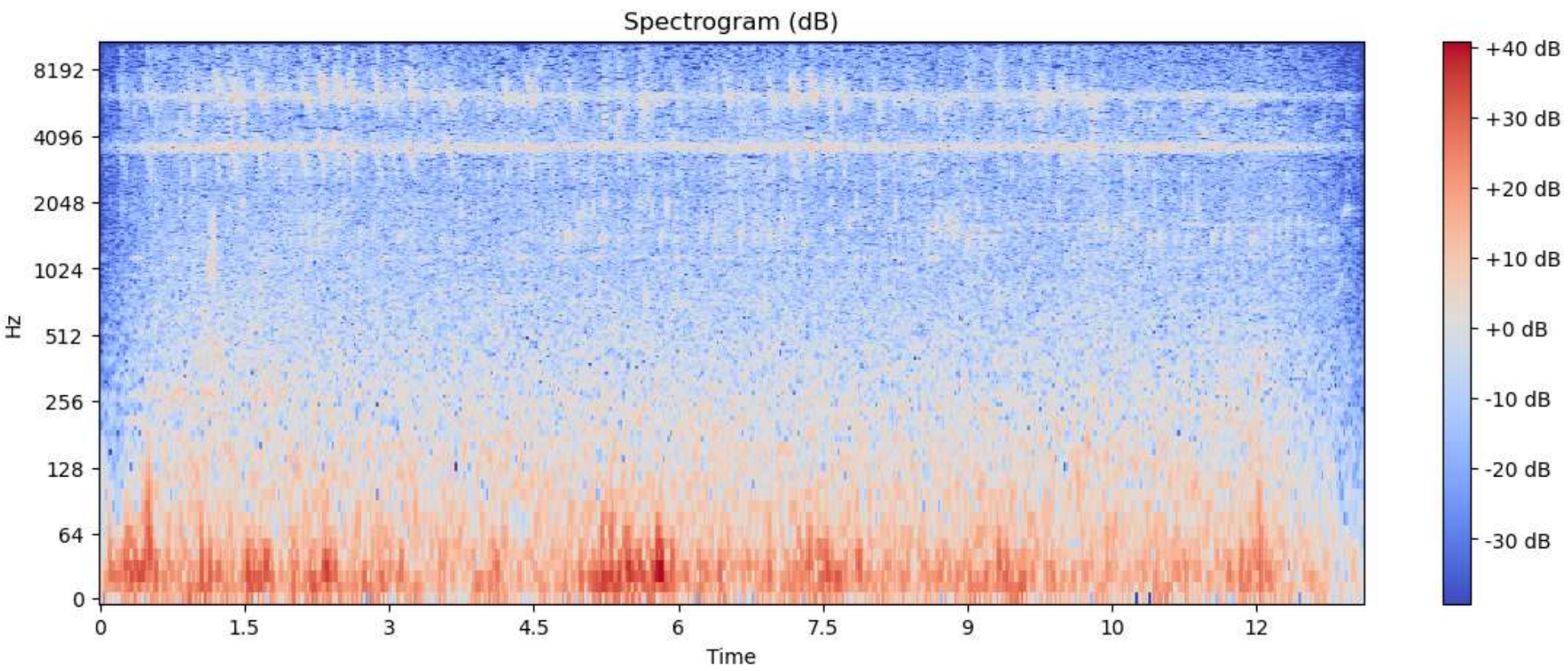
Scientific Name : Spermestes bicolor

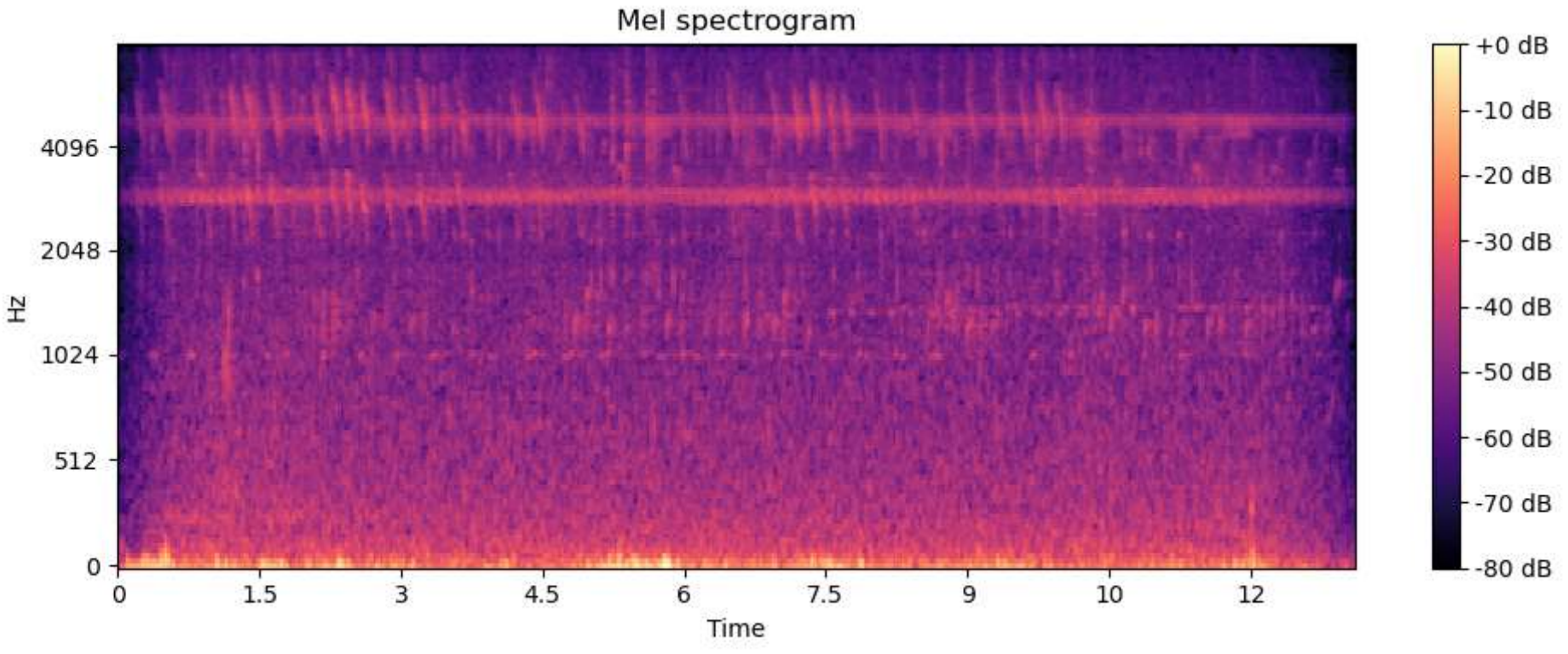
Top ↑

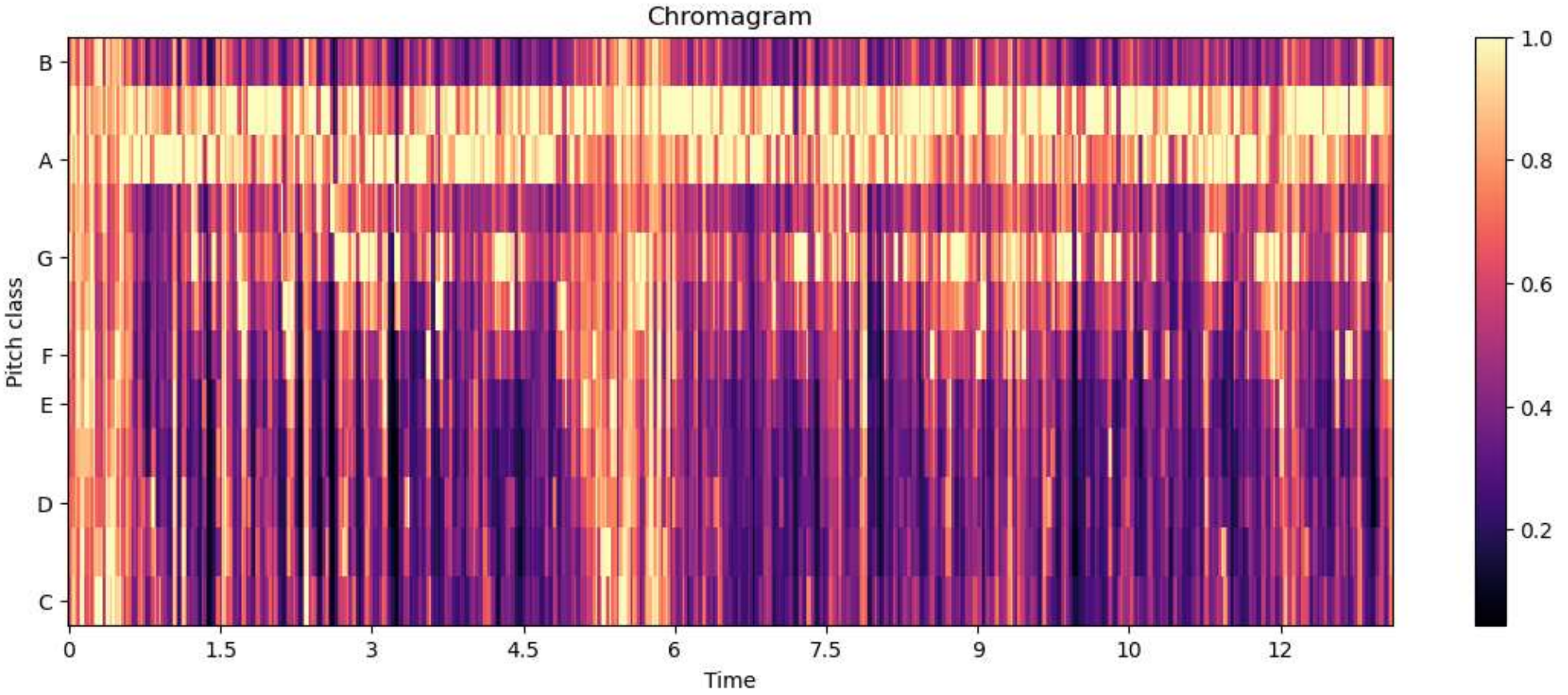
In [21]:
```
audio_eda("/kaggle/input/birdclef-2023/train_audio/bawman1/XC115075.ogg")
```

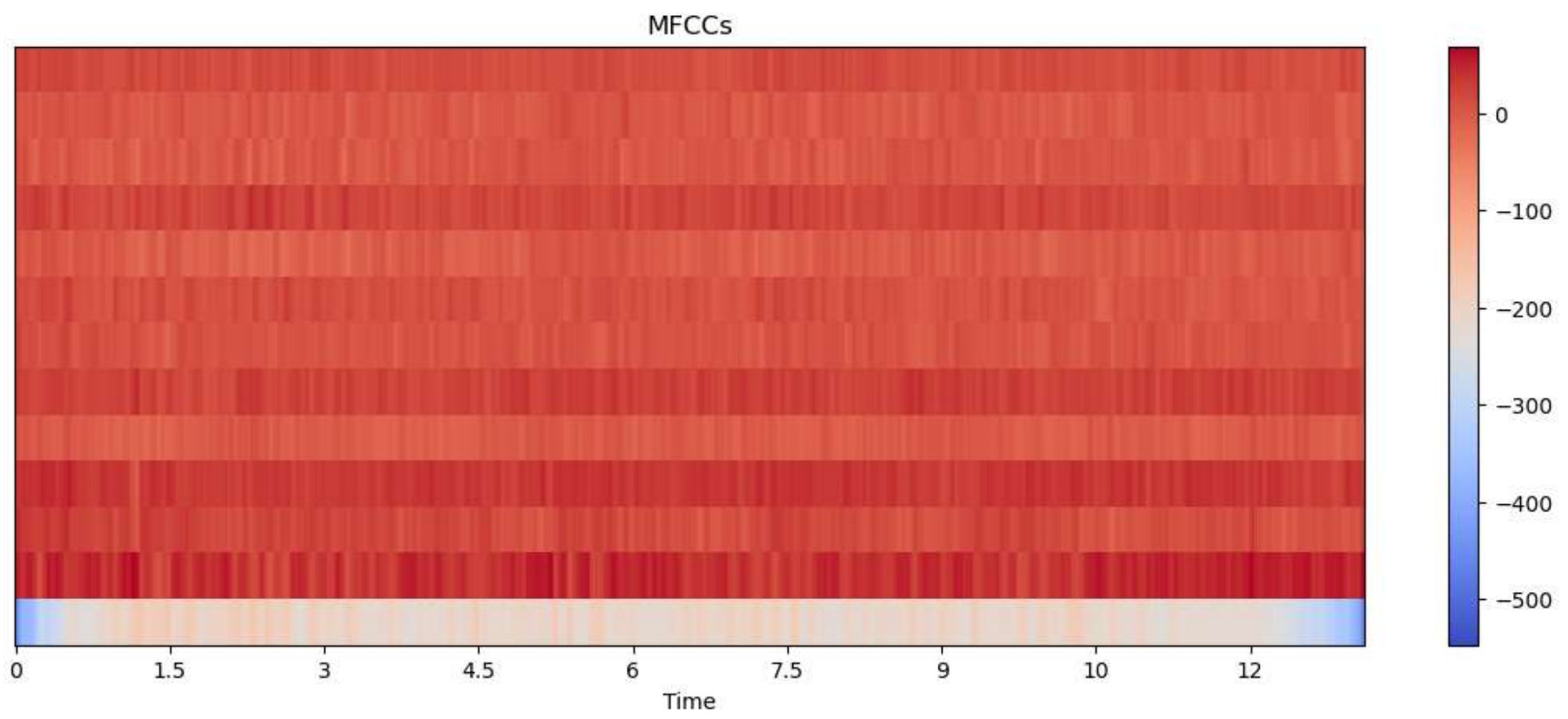
0:00 / 0:13


Waveform
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0
1.5
3
4.5
6
7.5
9
10
12
14
Time


Spectrogram (dB)
8192
4096
2048
1024
512
256
128
64
0
Hz
0
1.5
3
4.5
6
7.5
9
10
12
Time
+40 dB
+30 dB
+20 dB
+10 dB
+0 dB
-10 dB
-20 dB
-30 dB


Mel spectrogram
4096
2048
1024
512
0
Hz
0
1.5
3
4.5
6
7.5
9
10
12
Time
+0 dB
-10 dB
-20 dB
-30 dB
-40 dB
-50 dB
-60 dB
-70 dB
-80 dB





Thank You for reading 😊
Please do upvote if you liked the notebook
If you have any suggestions or feeback, please let me know

In [ ]: