EDA

In [2]:
```
# Import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

In [3]:
```
# List of datasets (based on the filiters in the proposal)
df_list = ['LT009.csv', 'LT010.csv', 'LT011.csv',
            'LT012.csv', 'LT014.csv', 'LT016.csv', 'LT017.csv',
            'LT018.csv', 'LT021.csv', 'SMB001.csv', 'SMB002.csv',
            'SMB005.csv', 'SMB006.csv', 'SMB007.csv',
            'SMB011.csv', 'SMB012.csv']
```

In [4]:
```
# Load all datasets into a dictionary
dataframes = {}
for dataset in df_list:
    var_name = dataset.replace(".csv", "")
    dataframes[var_name] = pd.read_csv(dataset)
```

In [43]:
```
# Combine all datasets into one DataFrame
df = pd.concat(dataframes.values(), ignore_index=True)
```

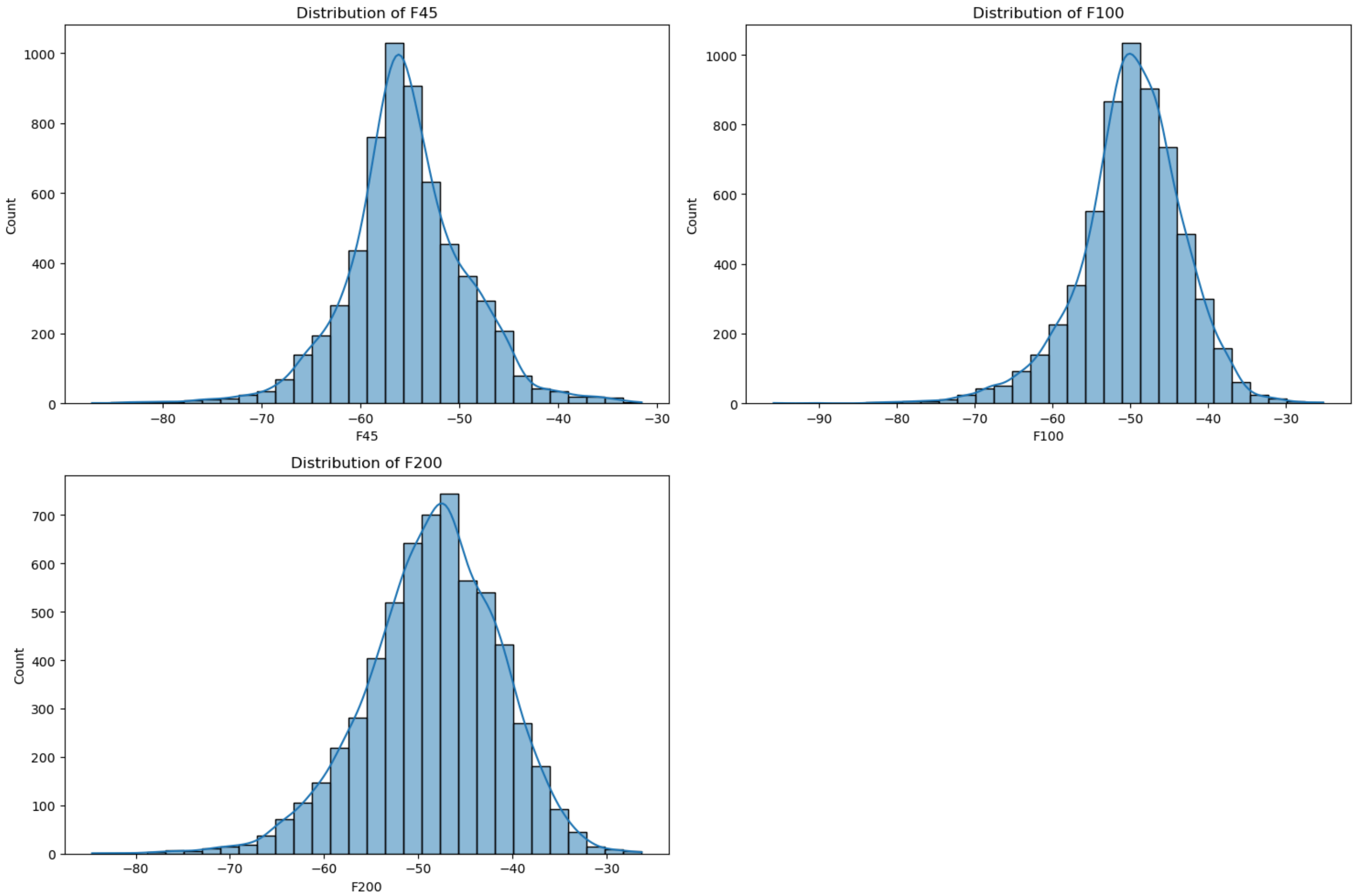
In [6]:
```
# Display the first few rows
print(df.head())

# Check for missing values
print(df.isnull().sum())

# Basic statistics
print(df.describe())

# Check the distribution of species
print(df['species'].value_counts())
```

```
  fishNum  dateSample        dateTimeSample dateProcessed    species  spCode  \
0   LT009  2022-07-26  2022-07-26T10:56:00Z    2022-07-26  lakeTrout      81
1   LT009  2022-07-26  2022-07-26T10:56:00Z    2022-07-26  lakeTrout      81
2   LT009  2022-07-26  2022-07-26T10:56:00Z    2022-07-26  lakeTrout      81
3   LT009  2022-07-26  2022-07-26T10:56:00Z    2022-07-26  lakeTrout      81
4   LT009  2022-07-26  2022-07-26T10:56:00Z    2022-07-26  lakeTrout      81

   totalLength  forkLength  weight  girth  ...     F255.5       F256  \
0          521         474    1132    236  ... -34.081596 -33.939062
1          521         474    1132    236  ... -37.477771 -38.291024
2          521         474    1132    236  ... -39.601506 -47.631764
3          521         474    1132    236  ... -46.987472 -47.106942
4          521         474    1132    236  ... -39.616575 -40.659007

      F256.5       F257     F257.5       F258     F258.5       F259  \
0 -34.633850 -37.575607 -40.060456 -35.903957 -34.051939 -34.936938
1 -43.002727 -42.791055 -36.913952 -35.235063 -37.331524 -42.207836
2 -54.888467 -46.378158 -47.450788 -57.292295 -57.364252 -55.097430
3 -41.711685 -38.940284 -39.023855 -40.670691 -42.545841 -45.924928
4 -43.263243 -45.481491 -43.871571 -41.978504 -42.040411 -42.759233

      F259.5       F260
0 -37.677605 -40.442006
1 -45.955729 -42.506945
2 -65.865182 -54.325638
3 -45.261969 -38.880901
4 -42.649847 -41.542901

[5 rows x 484 columns]
fishNum           0
dateSample        0
dateTimeSample    0
dateProcessed     0
species           0
                 ..
F258              0
F258.5            0
F259              0
F259.5            0
F260              0
Length: 484, dtype: int64
            spCode  totalLength   forkLength       weight       girth  \
count  6085.000000  6085.000000  6085.000000  6085.000000  6085.000000
mean    168.164339   490.677568   451.139195  1345.957272   274.821528
std     113.525837    67.177749    58.322325   354.544654    40.333961
min      81.000000   268.000000   252.000000   272.000000   170.000000
25%      81.000000   472.000000   432.000000  1278.000000   259.000000
50%      81.000000   503.000000   463.000000  1454.000000   270.000000
75%     316.000000   539.000000   486.000000  1636.000000   305.000000
max     316.000000   590.000000   547.000000  1944.000000   334.000000

       dorsoLatHeight          sex          mat  airbladderTotalLength  \
count     6085.000000  6085.000000  6085.000000            6085.000000
mean        50.071816     1.430074     1.988003             160.776171
std          8.962883     0.495127     0.108879              48.432149
min         22.000000     1.000000     1.000000              72.000000
25%         46.000000     1.000000     2.000000             104.000000
50%         53.000000     1.000000     2.000000             187.000000
75%         57.000000     2.000000     2.000000             199.000000
max         59.000000     2.000000     2.000000             225.000000

       airBladderWidth  ...       F255.5         F256       F256.5  \
count      6085.000000  ...  6085.000000  6085.000000  6085.000000
mean         33.959573  ...   -41.805443   -41.851171   -41.797916
std          12.212076  ...     7.070571     7.113147     7.099991
min          19.000000  ...   -78.843284   -82.577800   -79.661231
25%          25.000000  ...   -45.850017   -46.009660   -45.955636
50%          26.000000  ...   -41.163023   -41.272489   -41.331858
75%          48.000000  ...   -37.150114   -37.097257   -37.021694
max          60.000000  ...   -21.428864   -20.690171   -19.252406

              F257       F257.5         F258       F258.5         F259  \
count  6085.000000  6085.000000  6085.000000  6085.000000  6085.000000
```

```
mean     -41.672258   -41.656713   -41.479265   -41.684716   -41.894873
std        7.146349     7.316822     7.315413     7.488512     7.487082
min      -77.790637   -96.954163   -76.129517   -84.023077   -78.535368
25%      -45.946223   -46.061001   -45.963980   -46.051188   -46.465327
50%      -41.239244   -41.078815   -40.997623   -41.032659   -41.384480
75%      -36.905344   -36.841922   -36.549512   -36.529740   -36.716267
max      -18.894342   -17.663329   -15.665360   -15.139941   -15.748508

            F259.5         F260
count  6085.000000  6085.000000
mean    -42.401428   -42.781133
std       7.653336     7.452198
min     -86.357254   -97.657442
25%     -47.000432   -47.343505
50%     -41.833387   -42.237085
75%     -37.165801   -37.675192
max     -17.904206   -20.296669

[8 rows x 473 columns]
species
lakeTrout         3828
smallmouthBass    2257
Name: count, dtype: int64
```

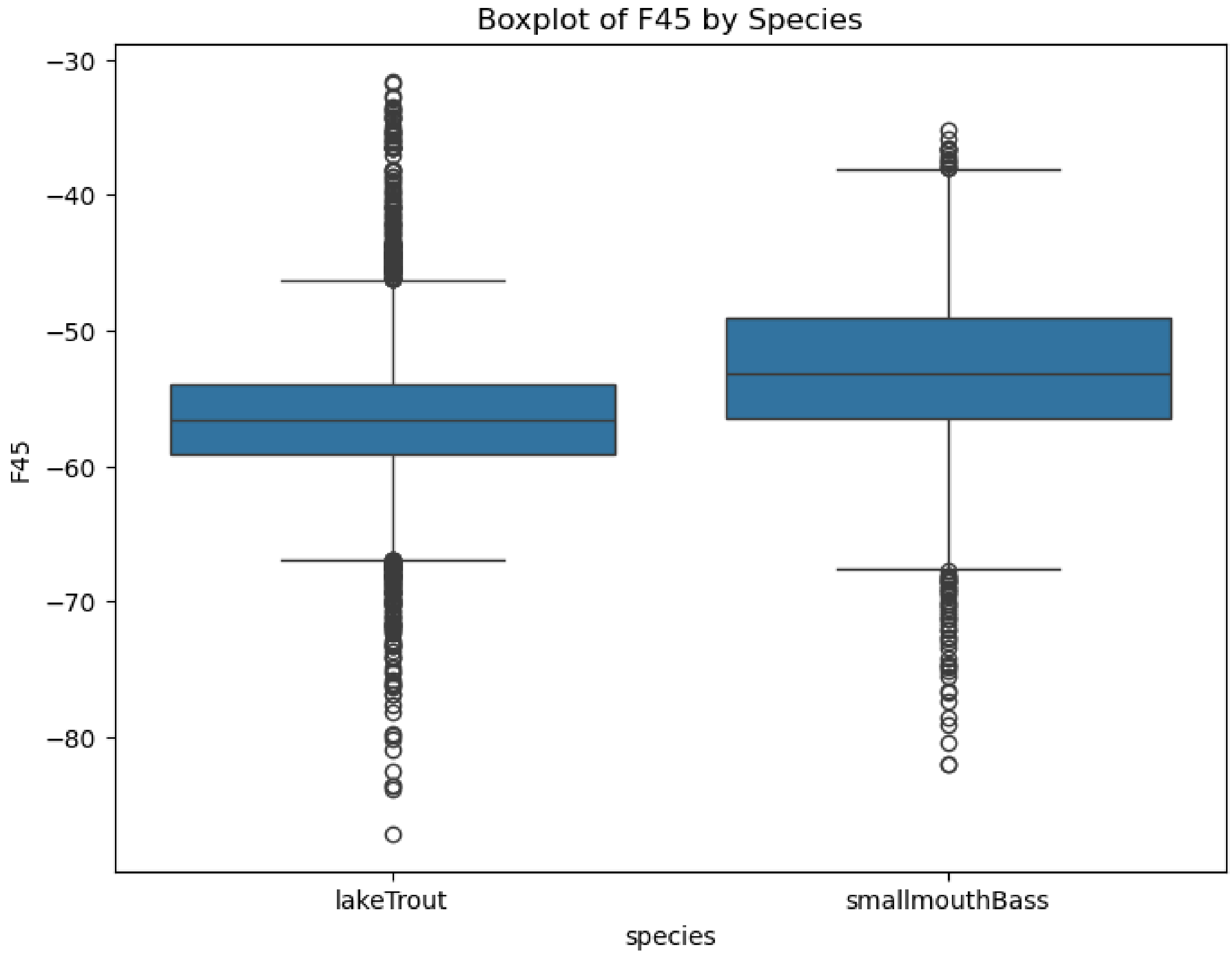
In [7]:

```python
# Plot histograms for a few frequencies (e.g., F45, F100, F200)
plt.figure(figsize=(15, 10))
for i, freq in enumerate(['F45', 'F100', 'F200']):
    plt.subplot(2, 2, i+1)
    sns.histplot(df[freq], kde=True, bins=30)
    plt.title(f'Distribution of {freq}')
plt.tight_layout()
plt.show()
```

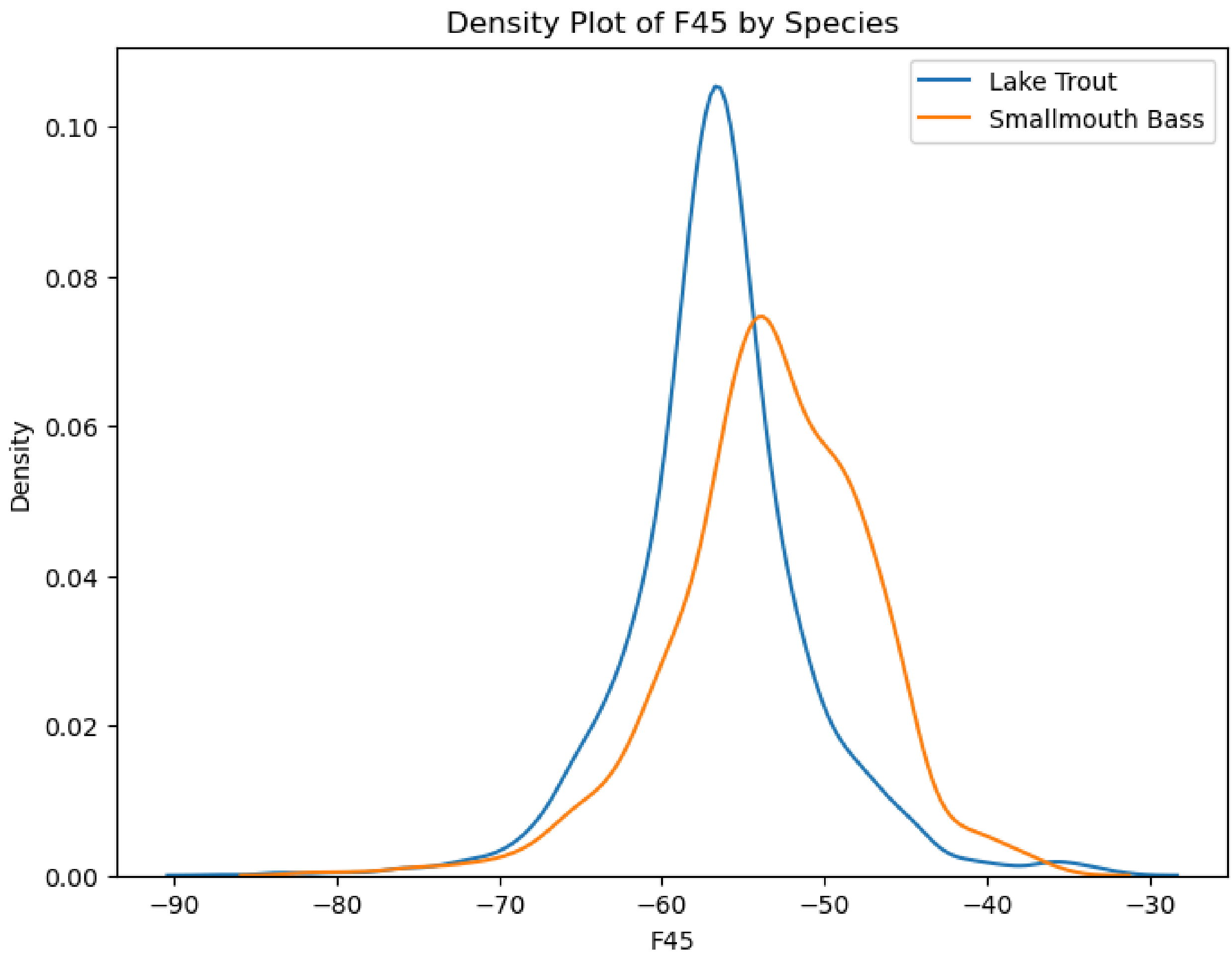


In [8]:

```python
# Boxplot for F45 by species
plt.figure(figsize=(8, 6))
sns.boxplot(x='species', y='F45', data=df)
plt.title('Boxplot of F45 by Species')
plt.show()
```



In [9]:
```
# Density plots for F45 by species
plt.figure(figsize=(8, 6))
sns.kdeplot(df[df['species'] == 'lakeTrout']['F45'], label='Lake Trout')
sns.kdeplot(df[df['species'] == 'smallmouthBass']['F45'], label='Smallmouth Bass')
plt.title('Density Plot of F45 by Species')
plt.legend()
plt.show()
```



In [10]:
```python
print(df.columns.tolist())
```

```
['fishNum', 'dateSample', 'dateTimeSample', 'dateProcessed', 'species', 'spCode', 'totalLength',
'forkLength', 'weight', 'girth', 'dorsoLatHeight', 'clipTag', 'sex', 'mat', 'airbladderTotalLengt
h', 'airBladderWidth', 'airbladderWeight', 'airBladderWeightCond', 'agingStructure', 'tissueSampl
e', 'Region_name', 'FishTrack', 'MaxTSdiff', 'Ping_time', 'deltaRange', 'deltaMinAng', 'deltaMajA
ng', 'aspectAngle', 'Target_range', 'Angle_minor_axis', 'Angle_major_axis', 'Distance_minor_axi
s', 'Distance_major_axis', 'StandDev_Angles_Minor_Axis', 'StandDev_Angles_Major_Axis', 'Target_tr
ue_depth', 'pingNumber', 'Ping_S', 'Ping_E', 'Num_targets', 'TS_mean', 'Target_range_mean', 'Spee
d_4D_mean_unsmoothed', 'Fish_track_change_in_range', 'Time_in_beam', 'Distance_3D_unsmoothed', 'T
hickness_mean', 'Exclude_below_line_range_mean', 'Target_depth_mean', 'Target_depth_max', 'Target
_depth_min', 'Fish_track_change_in_depth', 'Region_bottom_altitude_min', 'Region_bottom_altitude_
max', 'Region_bottom_altitude_mean', 'Region_top_altitude_min', 'Region_top_altitude_max', 'Regio
n_top_altitude_mean', 'F45', 'F45.5', 'F46', 'F46.5', 'F47', 'F47.5', 'F48', 'F48.5', 'F49', 'F4
9.5', 'F50', 'F50.5', 'F51', 'F51.5', 'F52', 'F52.5', 'F53', 'F53.5', 'F54', 'F54.5', 'F55', 'F5
5.5', 'F56', 'F56.5', 'F57', 'F57.5', 'F58', 'F58.5', 'F59', 'F59.5', 'F60', 'F60.5', 'F61', 'F6
1.5', 'F62', 'F62.5', 'F63', 'F63.5', 'F64', 'F64.5', 'F65', 'F65.5', 'F66', 'F66.5', 'F67', 'F6
7.5', 'F68', 'F68.5', 'F69', 'F69.5', 'F70', 'F70.5', 'F71', 'F71.5', 'F72', 'F72.5', 'F73', 'F7
3.5', 'F74', 'F74.5', 'F75', 'F75.5', 'F76', 'F76.5', 'F77', 'F77.5', 'F78', 'F78.5', 'F79', 'F7
9.5', 'F80', 'F80.5', 'F81', 'F81.5', 'F82', 'F82.5', 'F83', 'F83.5', 'F84', 'F84.5', 'F85', 'F8
5.5', 'F86', 'F86.5', 'F87', 'F87.5', 'F88', 'F88.5', 'F89', 'F89.5', 'F90', 'F90.5', 'F91', 'F9
1.5', 'F92', 'F92.5', 'F93', 'F93.5', 'F94', 'F94.5', 'F95', 'F95.5', 'F96', 'F96.5', 'F97', 'F9
7.5', 'F98', 'F98.5', 'F99', 'F99.5', 'F100', 'F100.5', 'F101', 'F101.5', 'F102', 'F102.5', 'F10
3', 'F103.5', 'F104', 'F104.5', 'F105', 'F105.5', 'F106', 'F106.5', 'F107', 'F107.5', 'F108', 'F1
08.5', 'F109', 'F109.5', 'F110', 'F110.5', 'F111', 'F111.5', 'F112', 'F112.5', 'F113', 'F113.5',
'F114', 'F114.5', 'F115', 'F115.5', 'F116', 'F116.5', 'F117', 'F117.5', 'F118', 'F118.5', 'F119',
'F119.5', 'F120', 'F120.5', 'F121', 'F121.5', 'F122', 'F122.5', 'F123', 'F123.5', 'F124', 'F124.
5', 'F125', 'F125.5', 'F126', 'F126.5', 'F127', 'F127.5', 'F128', 'F128.5', 'F129', 'F129.5', 'F1
30', 'F130.5', 'F131', 'F131.5', 'F132', 'F132.5', 'F133', 'F133.5', 'F134', 'F134.5', 'F135', 'F
135.5', 'F136', 'F136.5', 'F137', 'F137.5', 'F138', 'F138.5', 'F139', 'F139.5', 'F140', 'F140.5',
'F141', 'F141.5', 'F142', 'F142.5', 'F143', 'F143.5', 'F144', 'F144.5', 'F145', 'F145.5', 'F146',
'F146.5', 'F147', 'F147.5', 'F148', 'F148.5', 'F149', 'F149.5', 'F150', 'F150.5', 'F151', 'F151.
5', 'F152', 'F152.5', 'F153', 'F153.5', 'F154', 'F154.5', 'F155', 'F155.5', 'F156', 'F156.5', 'F1
57', 'F157.5', 'F158', 'F158.5', 'F159', 'F159.5', 'F160', 'F160.5', 'F161', 'F161.5', 'F162', 'F
162.5', 'F163', 'F163.5', 'F164', 'F164.5', 'F165', 'F165.5', 'F166', 'F166.5', 'F167', 'F167.5',
'F168', 'F168.5', 'F169', 'F169.5', 'F170', 'F173', 'F173.5', 'F174', 'F174.5', 'F175', 'F175.5',
'F176', 'F176.5', 'F177', 'F177.5', 'F178', 'F178.5', 'F179', 'F179.5', 'F180', 'F180.5', 'F181',
'F181.5', 'F182', 'F182.5', 'F183', 'F183.5', 'F184', 'F184.5', 'F185', 'F185.5', 'F186', 'F186.
5', 'F187', 'F187.5', 'F188', 'F188.5', 'F189', 'F189.5', 'F190', 'F190.5', 'F191', 'F191.5', 'F1
92', 'F192.5', 'F193', 'F193.5', 'F194', 'F194.5', 'F195', 'F195.5', 'F196', 'F196.5', 'F197', 'F
197.5', 'F198', 'F198.5', 'F199', 'F199.5', 'F200', 'F200.5', 'F201', 'F201.5', 'F202', 'F202.5',
'F203', 'F203.5', 'F204', 'F204.5', 'F205', 'F205.5', 'F206', 'F206.5', 'F207', 'F207.5', 'F208',
'F208.5', 'F209', 'F209.5', 'F210', 'F210.5', 'F211', 'F211.5', 'F212', 'F212.5', 'F213', 'F213.
5', 'F214', 'F214.5', 'F215', 'F215.5', 'F216', 'F216.5', 'F217', 'F217.5', 'F218', 'F218.5', 'F2
19', 'F219.5', 'F220', 'F220.5', 'F221', 'F221.5', 'F222', 'F222.5', 'F223', 'F223.5', 'F224', 'F
224.5', 'F225', 'F225.5', 'F226', 'F226.5', 'F227', 'F227.5', 'F228', 'F228.5', 'F229', 'F229.5',
'F230', 'F230.5', 'F231', 'F231.5', 'F232', 'F232.5', 'F233', 'F233.5', 'F234', 'F234.5', 'F235',
'F235.5', 'F236', 'F236.5', 'F237', 'F237.5', 'F238', 'F238.5', 'F239', 'F239.5', 'F240', 'F240.
5', 'F241', 'F241.5', 'F242', 'F242.5', 'F243', 'F243.5', 'F244', 'F244.5', 'F245', 'F245.5', 'F2
46', 'F246.5', 'F247', 'F247.5', 'F248', 'F248.5', 'F249', 'F249.5', 'F250', 'F250.5', 'F251', 'F
251.5', 'F252', 'F252.5', 'F253', 'F253.5', 'F254', 'F254.5', 'F255', 'F255.5', 'F256', 'F256.5',
'F257', 'F257.5', 'F258', 'F258.5', 'F259', 'F259.5', 'F260']
```

In [44]:
```python
# Convert 'dateProcessed' to proper datetime
df["dateProcessed"] = pd.to_datetime(df["dateProcessed"])

# Convert 'Ping_time' to a time format
df["Ping_time"] = pd.to_datetime(df["Ping_time"].str.strip(), format="%H:%M:%S.%f").dt.time

# Merge dateProcessed (date) and Ping_time (time) into one datetime column
df["Ping_time"] = df.apply(lambda row: pd.Timestamp.combine(row["dateProcessed"], row["Ping_time"

# Verify output
print(df[["dateProcessed", "Ping_time"]])
```

```
      dateProcessed               Ping_time
0        2022-07-26 2022-07-26 15:01:17.016
1        2022-07-26 2022-07-26 15:01:17.220
2        2022-07-26 2022-07-26 15:01:19.119
3        2022-07-26 2022-07-26 15:01:19.220
4        2022-07-26 2022-07-26 15:04:37.217
...             ...                     ...
6080     2022-07-28 2022-07-28 00:17:22.964
6081     2022-07-28 2022-07-28 00:17:25.964
6082     2022-07-28 2022-07-28 00:17:26.163
6083     2022-07-28 2022-07-28 00:17:40.564
6084     2022-07-28 2022-07-28 00:17:40.764

[6085 rows x 2 columns]
```

In [45]:

```
# Filter for the specific fish (e.g., LT009)
df_LT009 = df[df['fishNum'] == 'LT009']

# Create the interactive plot
fig = go.Figure()

fig.add_trace(go.Scatter(
    x=df_LT009['Ping_time'],
    y=df_LT009['F45'],
    mode='markers',  # Markers help visualize missing points
    name='F45',
    connectgaps=False,  # Ensures missing time points are NOT connected
    hoverinfo="skip",  # Disable default hover
    hovertemplate="Time: %{x|%H:%M:%S.%f}<br>F45: %{y}<extra></extra>"  # Custom hover format
))

# Add interactive time slider and range selector
fig.update_layout(
    title="F45 Over Time for LT009 (Natural Gaps)",
    xaxis_title="Time",
    yaxis_title="F45 Response",
    xaxis=dict(
        rangeselector=dict(
            buttons=[
                dict(count=10, label="10 mins", step="minute", stepmode="backward"),
                dict(count=30, label="30 mins", step="minute", stepmode="backward"),
                dict(step="all")
            ]
        ),
        rangeslider=dict(visible=True),  # Interactive sliding window
        type="date"
    ),
    template="plotly_white"
)
# Show the figure
fig.show()
```

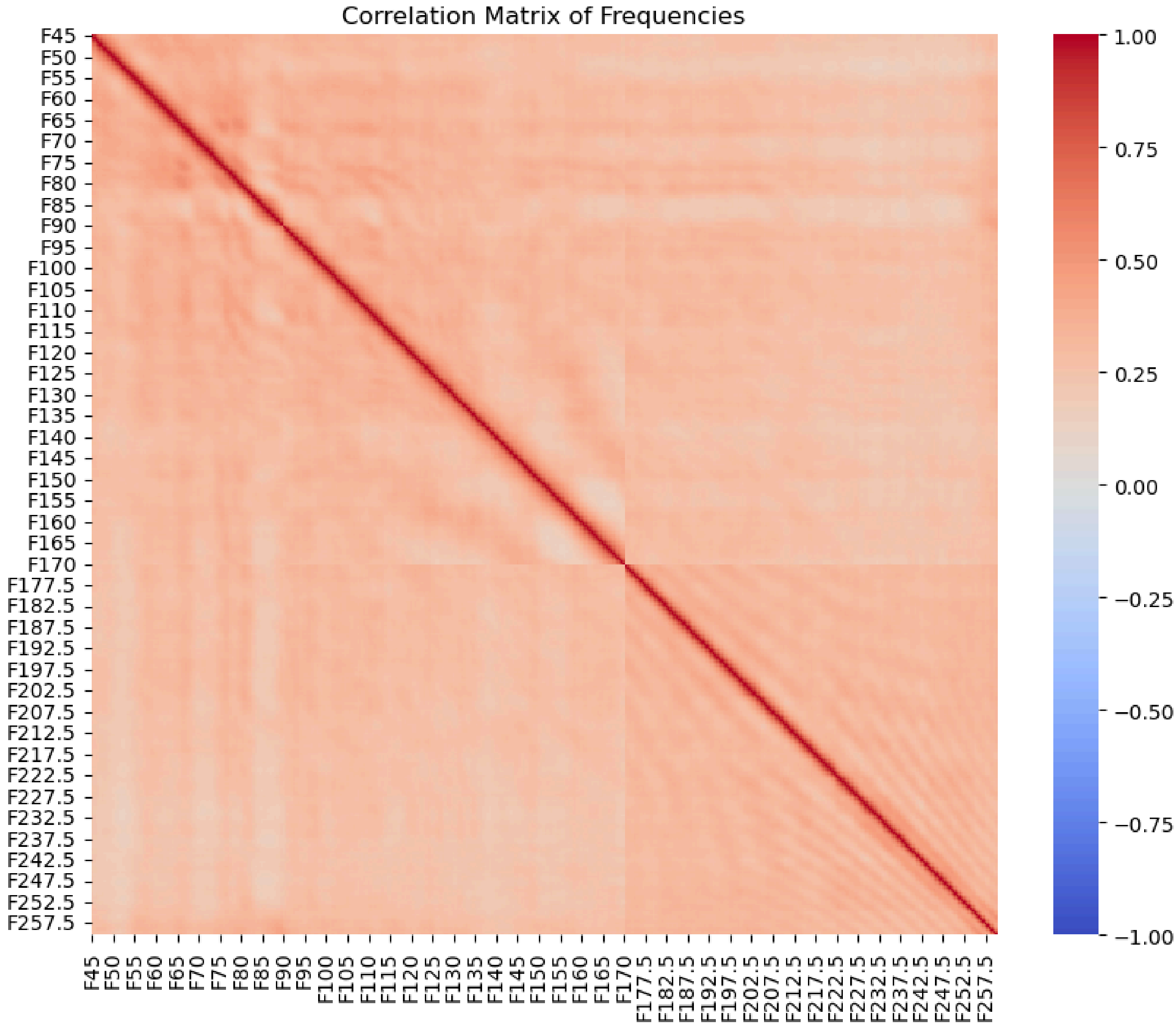
In [47]:

```
import re

# Example list of column names
columns = df.columns.tolist()

# Use regex to find columns starting with 'F' followed by numbers
f_columns = [col for col in columns if re.match(r"^F\d+(\.\d+)?$", col)]

# Print the result
print(f_columns)
```

```
['F45', 'F45.5', 'F46', 'F46.5', 'F47', 'F47.5', 'F48', 'F48.5', 'F49', 'F49.5', 'F50', 'F50.5',
'F51', 'F51.5', 'F52', 'F52.5', 'F53', 'F53.5', 'F54', 'F54.5', 'F55', 'F55.5', 'F56', 'F56.5',
'F57', 'F57.5', 'F58', 'F58.5', 'F59', 'F59.5', 'F60', 'F60.5', 'F61', 'F61.5', 'F62', 'F62.5',
'F63', 'F63.5', 'F64', 'F64.5', 'F65', 'F65.5', 'F66', 'F66.5', 'F67', 'F67.5', 'F68', 'F68.5',
'F69', 'F69.5', 'F70', 'F70.5', 'F71', 'F71.5', 'F72', 'F72.5', 'F73', 'F73.5', 'F74', 'F74.5',
'F75', 'F75.5', 'F76', 'F76.5', 'F77', 'F77.5', 'F78', 'F78.5', 'F79', 'F79.5', 'F80', 'F80.5',
'F81', 'F81.5', 'F82', 'F82.5', 'F83', 'F83.5', 'F84', 'F84.5', 'F85', 'F85.5', 'F86', 'F86.5',
'F87', 'F87.5', 'F88', 'F88.5', 'F89', 'F89.5', 'F90', 'F90.5', 'F91', 'F91.5', 'F92', 'F92.5',
'F93', 'F93.5', 'F94', 'F94.5', 'F95', 'F95.5', 'F96', 'F96.5', 'F97', 'F97.5', 'F98', 'F98.5',
'F99', 'F99.5', 'F100', 'F100.5', 'F101', 'F101.5', 'F102', 'F102.5', 'F103', 'F103.5', 'F104',
'F104.5', 'F105', 'F105.5', 'F106', 'F106.5', 'F107', 'F107.5', 'F108', 'F108.5', 'F109', 'F109.
5', 'F110', 'F110.5', 'F111', 'F111.5', 'F112', 'F112.5', 'F113', 'F113.5', 'F114', 'F114.5', 'F1
15', 'F115.5', 'F116', 'F116.5', 'F117', 'F117.5', 'F118', 'F118.5', 'F119', 'F119.5', 'F120', 'F
120.5', 'F121', 'F121.5', 'F122', 'F122.5', 'F123', 'F123.5', 'F124', 'F124.5', 'F125', 'F125.5',
'F126', 'F126.5', 'F127', 'F127.5', 'F128', 'F128.5', 'F129', 'F129.5', 'F130', 'F130.5', 'F131',
'F131.5', 'F132', 'F132.5', 'F133', 'F133.5', 'F134', 'F134.5', 'F135', 'F135.5', 'F136', 'F136.
5', 'F137', 'F137.5', 'F138', 'F138.5', 'F139', 'F139.5', 'F140', 'F140.5', 'F141', 'F141.5', 'F1
42', 'F142.5', 'F143', 'F143.5', 'F144', 'F144.5', 'F145', 'F145.5', 'F146', 'F146.5', 'F147', 'F
147.5', 'F148', 'F148.5', 'F149', 'F149.5', 'F150', 'F150.5', 'F151', 'F151.5', 'F152', 'F152.5',
'F153', 'F153.5', 'F154', 'F154.5', 'F155', 'F155.5', 'F156', 'F156.5', 'F157', 'F157.5', 'F158',
'F158.5', 'F159', 'F159.5', 'F160', 'F160.5', 'F161', 'F161.5', 'F162', 'F162.5', 'F163', 'F163.
5', 'F164', 'F164.5', 'F165', 'F165.5', 'F166', 'F166.5', 'F167', 'F167.5', 'F168', 'F168.5', 'F1
69', 'F169.5', 'F170', 'F173', 'F173.5', 'F174', 'F174.5', 'F175', 'F175.5', 'F176', 'F176.5', 'F
177', 'F177.5', 'F178', 'F178.5', 'F179', 'F179.5', 'F180', 'F180.5', 'F181', 'F181.5', 'F182',
'F182.5', 'F183', 'F183.5', 'F184', 'F184.5', 'F185', 'F185.5', 'F186', 'F186.5', 'F187', 'F187.
5', 'F188', 'F188.5', 'F189', 'F189.5', 'F190', 'F190.5', 'F191', 'F191.5', 'F192', 'F192.5', 'F1
93', 'F193.5', 'F194', 'F194.5', 'F195', 'F195.5', 'F196', 'F196.5', 'F197', 'F197.5', 'F198', 'F
198.5', 'F199', 'F199.5', 'F200', 'F200.5', 'F201', 'F201.5', 'F202', 'F202.5', 'F203', 'F203.5',
'F204', 'F204.5', 'F205', 'F205.5', 'F206', 'F206.5', 'F207', 'F207.5', 'F208', 'F208.5', 'F209',
'F209.5', 'F210', 'F210.5', 'F211', 'F211.5', 'F212', 'F212.5', 'F213', 'F213.5', 'F214', 'F214.
5', 'F215', 'F215.5', 'F216', 'F216.5', 'F217', 'F217.5', 'F218', 'F218.5', 'F219', 'F219.5', 'F2
20', 'F220.5', 'F221', 'F221.5', 'F222', 'F222.5', 'F223', 'F223.5', 'F224', 'F224.5', 'F225', 'F
225.5', 'F226', 'F226.5', 'F227', 'F227.5', 'F228', 'F228.5', 'F229', 'F229.5', 'F230', 'F230.5',
'F231', 'F231.5', 'F232', 'F232.5', 'F233', 'F233.5', 'F234', 'F234.5', 'F235', 'F235.5', 'F236',
'F236.5', 'F237', 'F237.5', 'F238', 'F238.5', 'F239', 'F239.5', 'F240', 'F240.5', 'F241', 'F241.
5', 'F242', 'F242.5', 'F243', 'F243.5', 'F244', 'F244.5', 'F245', 'F245.5', 'F246', 'F246.5', 'F2
47', 'F247.5', 'F248', 'F248.5', 'F249', 'F249.5', 'F250', 'F250.5', 'F251', 'F251.5', 'F252', 'F
252.5', 'F253', 'F253.5', 'F254', 'F254.5', 'F255', 'F255.5', 'F256', 'F256.5', 'F257', 'F257.5',
'F258', 'F258.5', 'F259', 'F259.5', 'F260']
```

In [63]:

```
# Compute the correlation matrix
corr_matrix = df[f_columns].corr().fillna(0)  # Fill NaNs if any

# Plot the heatmap with correct color scaling
plt.figure(figsize=(10, 8))
plt.style.use('default')  # Ensure no grayscale styles are applied
sns.heatmap(corr_matrix, annot=False, cmap='coolwarm', vmin=-1, vmax=1)  # Force correct color sc
plt.title('Correlation Matrix of Frequencies')
plt.show()
```



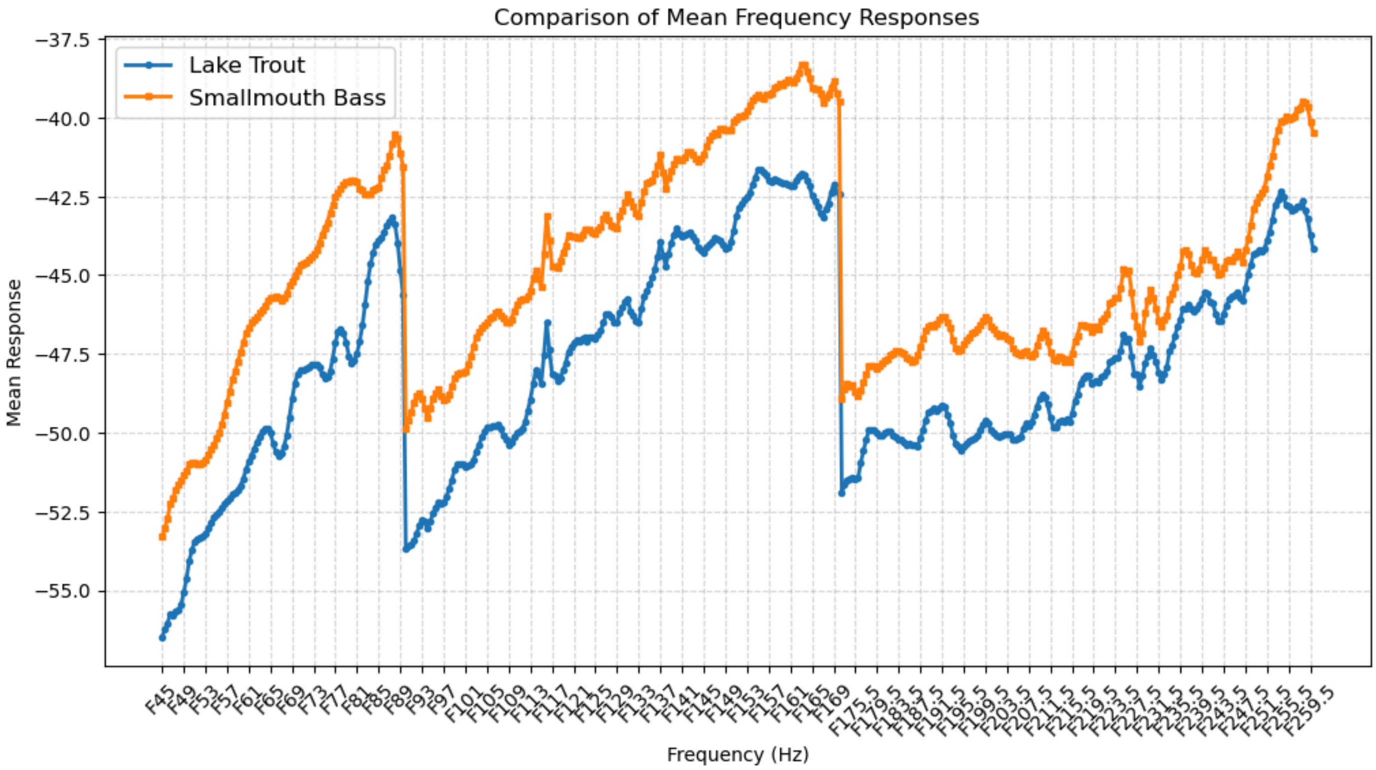
In [67]:
```
# Split the dataset by species
lt_df = df[df['species'] == 'lakeTrout']
smb_df = df[df['species'] == 'smallmouthBass']

# Check the size of each dataset
print(f"Lake Trout dataset size: {lt_df.shape}")
print(f"Smallmouth Bass dataset size: {smb_df.shape}")
```

```
Lake Trout dataset size: (3828, 484)
Smallmouth Bass dataset size: (2257, 484)
```

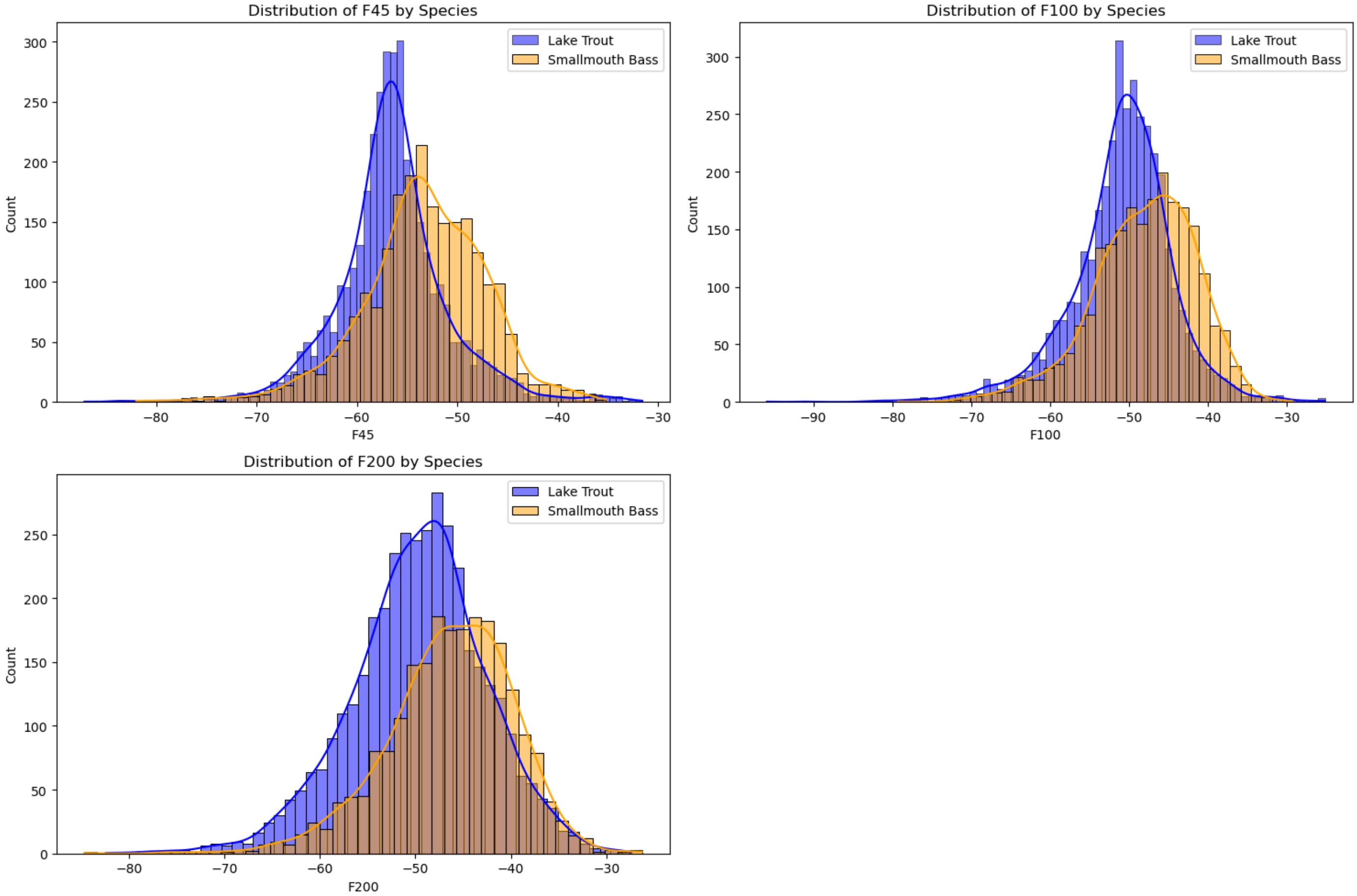
In [73]:
```
# Compare mean frequency responses
lt_mean = lt_df[f_columns].mean()  # Assuming frequencies start from column 4
smb_mean = smb_df[f_columns].mean()

# Create a DataFrame for comparison
mean_comparison = pd.DataFrame({'Lake Trout': lt_mean, 'Smallmouth Bass': smb_mean})
print(mean_comparison)

# Define figure size
plt.figure(figsize=(12, 6))

# Reduce markers (plot every nth point to avoid clutter)
n = max(1, len(mean_comparison) // 50)  # Adjust dynamically

# Plot mean frequency responses
plt.plot(mean_comparison.index, mean_comparison['Lake Trout'], label="Lake Trout", marker='o', ma
plt.plot(mean_comparison.index, mean_comparison['Smallmouth Bass'], label="Smallmouth Bass", marl

# Formatting improvements
plt.xlabel("Frequency (Hz)")
plt.ylabel("Mean Response")
plt.title("Comparison of Mean Frequency Responses")
```

```
# Reduce x-axis ticks for readability
plt.xticks(mean_comparison.index[::n], rotation=45, fontsize=10)  # Plot every nth tick

# Use a less intense grid
plt.grid(True, linestyle="--", alpha=0.5)

# Add a legend with better styling
plt.legend(frameon=True, loc="upper left", fontsize=12)

# Show plot
plt.show()
```

```
        Lake Trout  Smallmouth Bass
F45     -56.490225       -53.302915
F45.5   -56.244414       -53.013019
F46     -56.064421       -52.730495
F46.5   -55.753755       -52.259890
F47     -55.807013       -52.095333
...            ...              ...
F258    -42.646562       -39.499465
F258.5  -42.949598       -39.539405
F259    -43.211695       -39.661469
F259.5  -43.725793       -40.155230
F260    -44.149846       -40.459719

[426 rows x 2 columns]
```

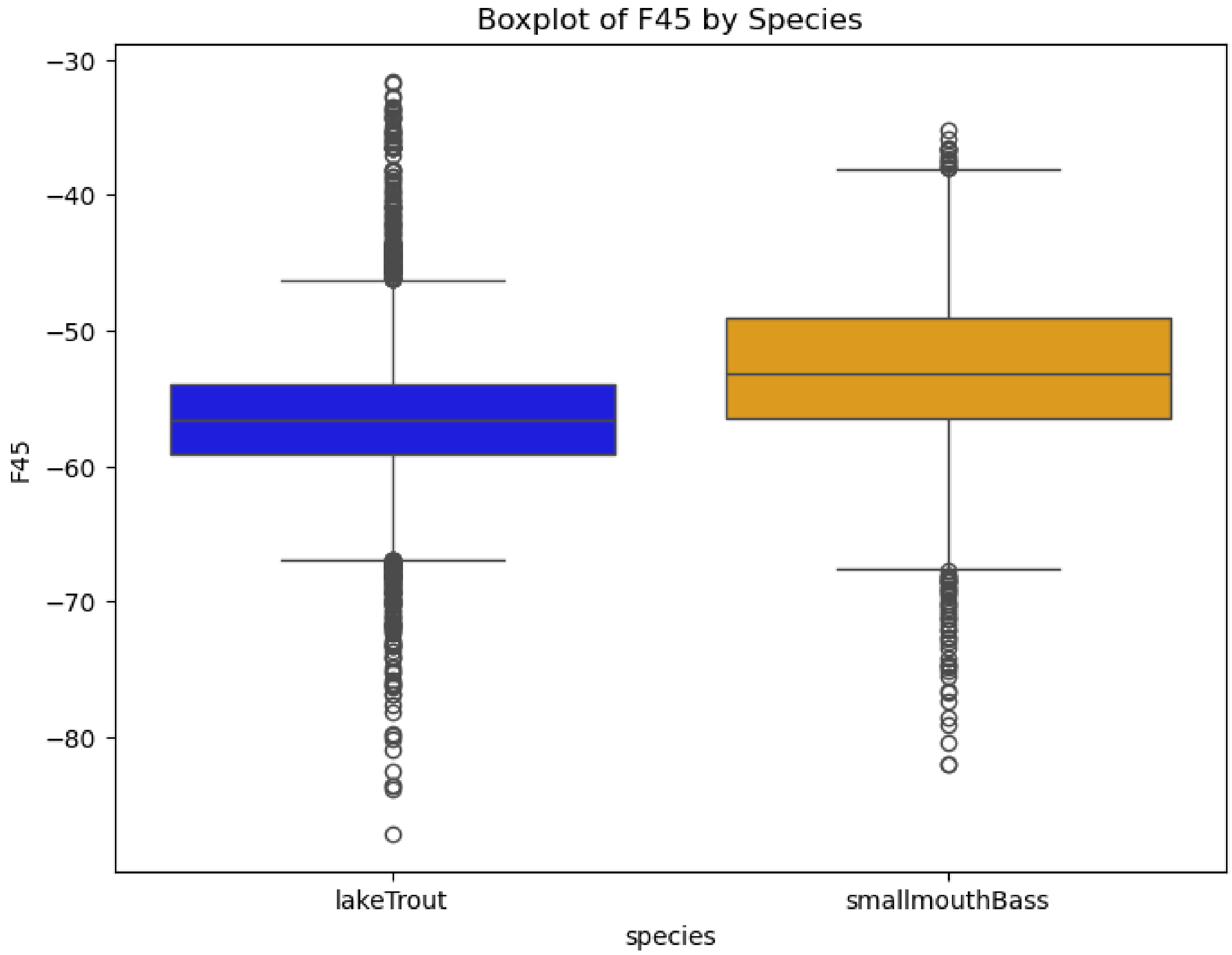


In [74]:
```
# Plot histograms for a few frequencies (e.g., F45, F100, F200)
frequencies = ['F45', 'F100', 'F200']

plt.figure(figsize=(15, 10))
for i, freq in enumerate(frequencies):
    plt.subplot(2, 2, i+1)
    sns.histplot(lt_df[freq], kde=True, color='blue', label='Lake Trout')
    sns.histplot(smb_df[freq], kde=True, color='orange', label='Smallmouth Bass')
    plt.title(f'Distribution of {freq} by Species')
    plt.legend()
plt.tight_layout()
plt.show()
```

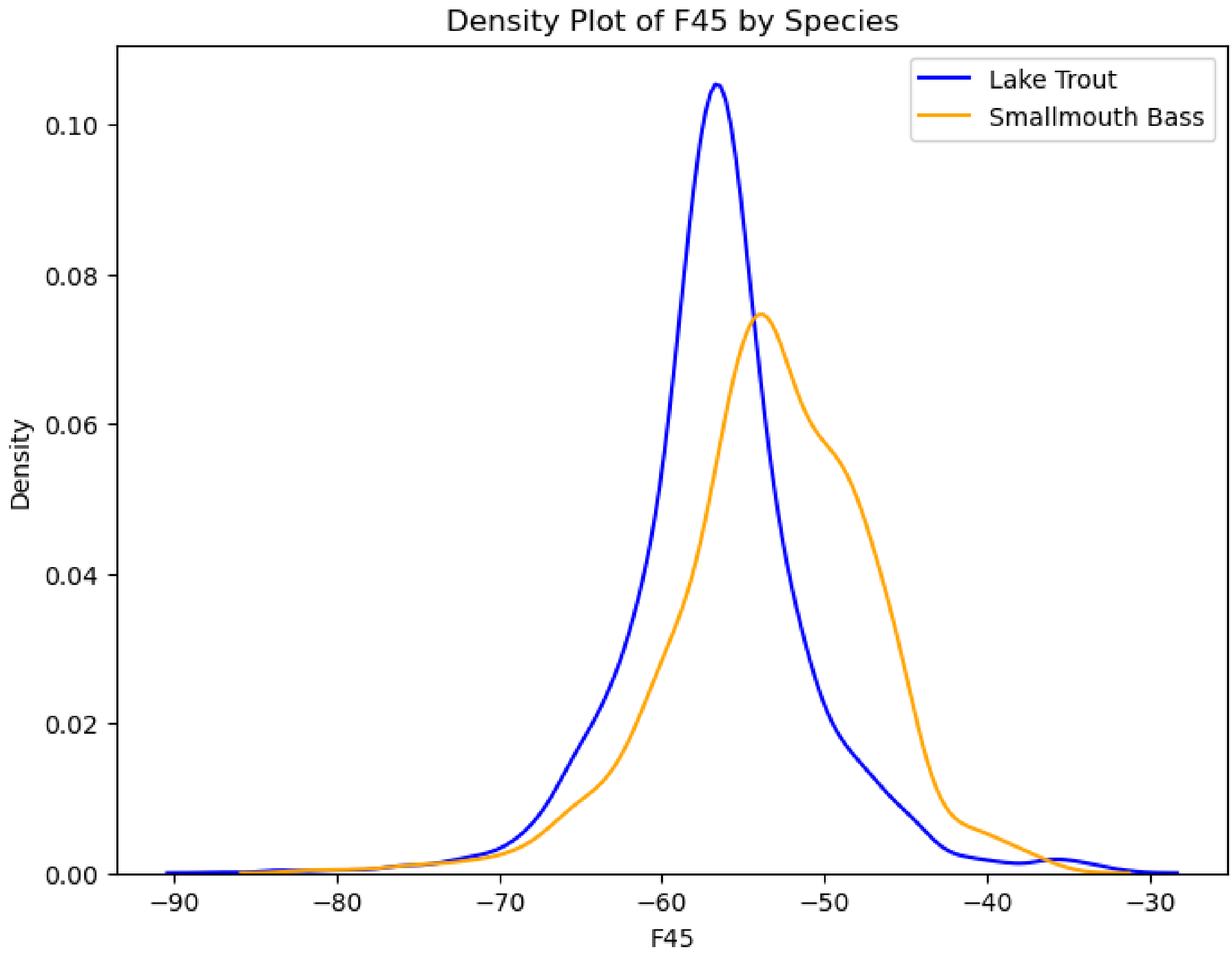


In [76]:
```
# Boxplot for F45 by species
plt.figure(figsize=(8, 6))
sns.boxplot(x='species', y='F45', data=df, palette=['blue', 'orange'])
plt.title('Boxplot of F45 by Species')
plt.show()
```

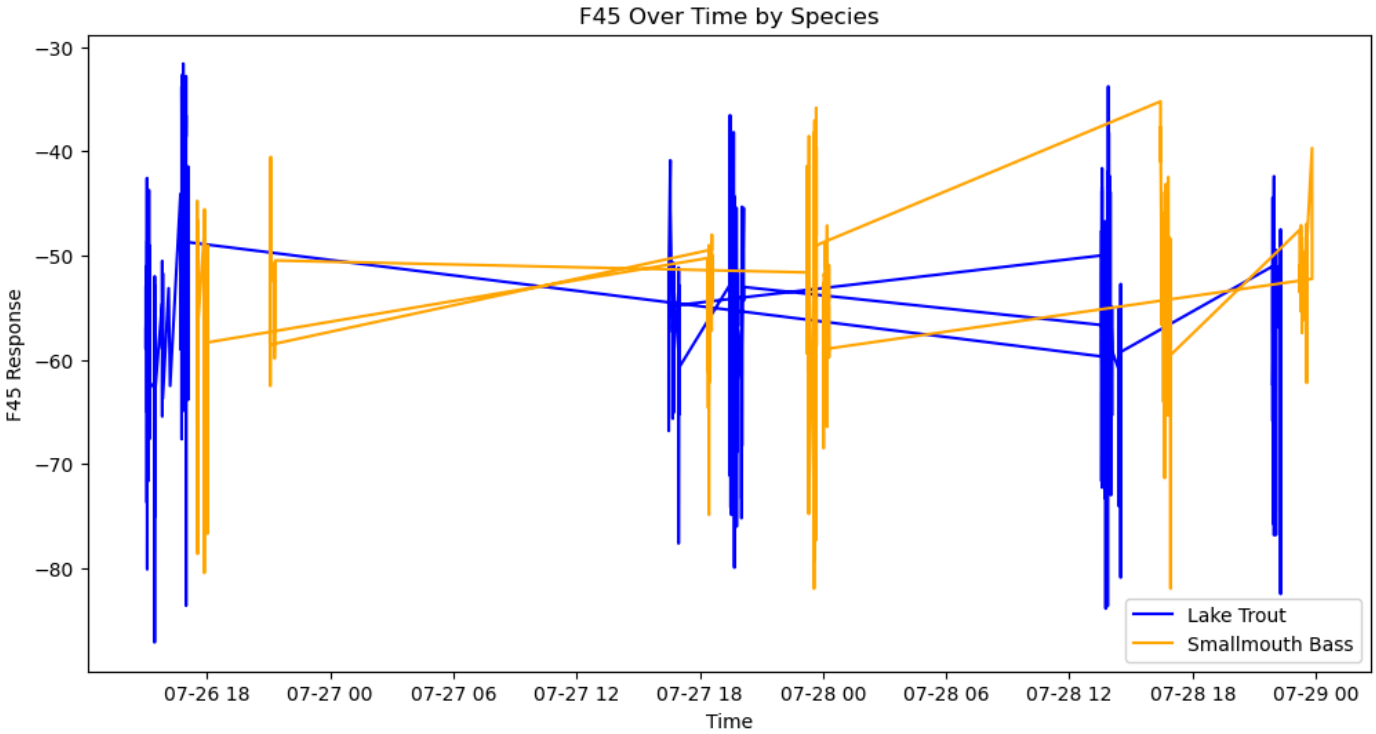
```
/var/folders/46/_qfvxk655mzdxw5h9v4z7ybw0000gn/T/ipykernel_54629/27757218.py:3: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign th
e `x` variable to `hue` and set `legend=False` for the same effect.
```



In [77]:
```
# Density plots for F45 by species
plt.figure(figsize=(8, 6))
sns.kdeplot(lt_df['F45'], label='Lake Trout', color='blue')
sns.kdeplot(smb_df['F45'], label='Smallmouth Bass', color='orange')
plt.title('Density Plot of F45 by Species')
plt.legend()
plt.show()
```



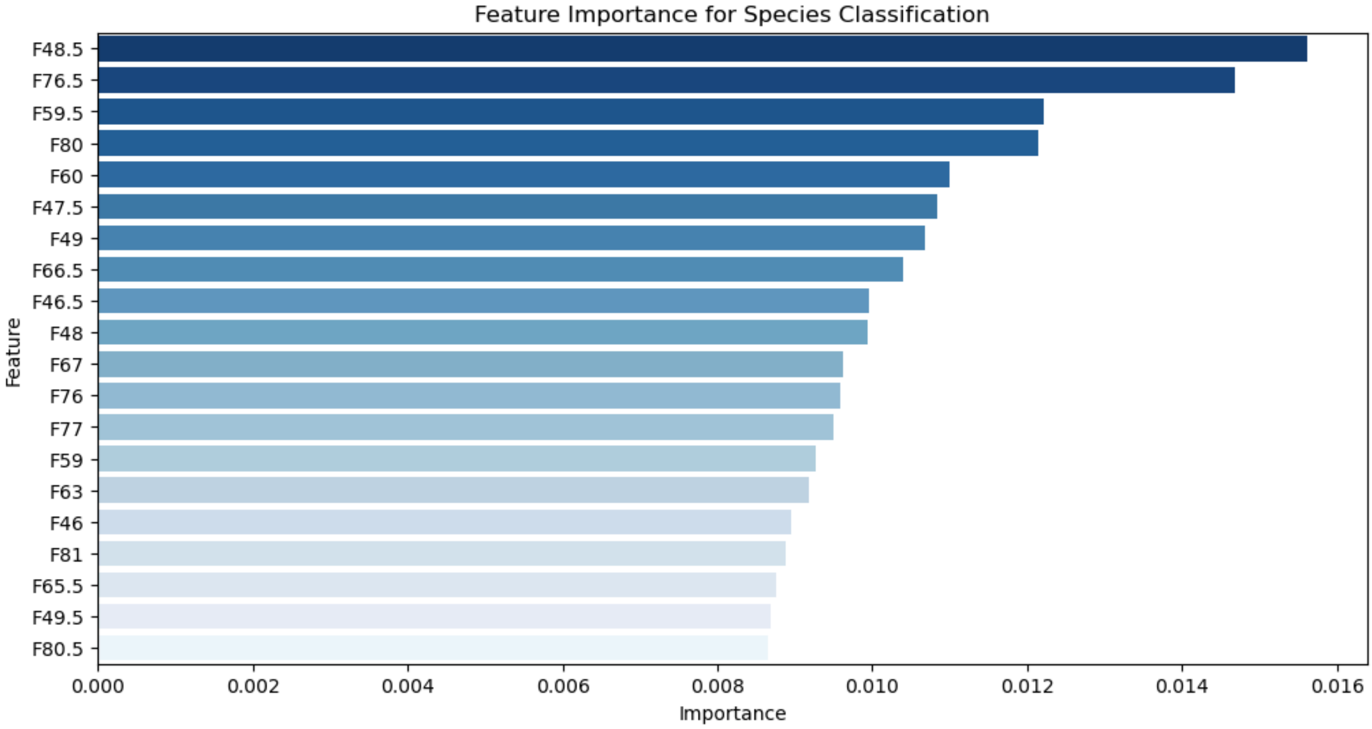
In [78]:
```
# Plot F45 over time for both species
plt.figure(figsize=(12, 6))
plt.plot(lt_df['Ping_time'], lt_df['F45'], label='Lake Trout', color='blue')
plt.plot(smb_df['Ping_time'], smb_df['F45'], label='Smallmouth Bass', color='orange')
plt.title('F45 Over Time by Species')
plt.xlabel('Time')
plt.ylabel('F45 Response')
plt.legend()
plt.show()
```



In [84]:
```
# Merge both datasets for classification
X = pd.concat([lt_df[f_columns], smb_df[f_columns]], axis=0)
y = pd.concat([lt_df['species'], smb_df['species']], axis=0)

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Random Forest Model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Feature Importances
importances = model.feature_importances_
feature_names = X.columns
```

In [88]:
```
# Create DataFrame for Feature Importances
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})

# Sort Features by Importance
feature_importance_df = feature_importance_df.sort_values(by="Importance", ascending=False)

# Plot Feature Importance
plt.figure(figsize=(12, 6))
sns.barplot(data=feature_importance_df[:20], x="Importance", y="Feature", palette="Blues_r")
plt.title('Feature Importance for Species Classification')
plt.show()
```

```
/var/folders/46/_qfvxk655mzdxw5h9v4z7ybw0000gn/T/ipykernel_54629/3877114549.py:9: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign th
e `y` variable to `hue` and set `legend=False` for the same effect.
```


Feature Importance for Species Classification
Feature
F48.5
F76.5
F59.5
F80
F60
F47.5
F49
F66.5
F46.5
F48
F67
F76
F77
F59
F63
F46
F81
F65.5
F49.5
F80.5
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0.016
Importance