In [39]:
```
#Import required libraries
import pandas as pd
import numpy as np
!pip install sqlalchemy psycopg2-binary
import warnings
warnings.filterwarnings('ignore')
```

```
Requirement already satisfied: sqlalchemy in c:\users\anuradha\anaconda3\lib\site-pa
ckages (2.0.34)
Collecting psycopg2-binary
  Downloading psycopg2_binary-2.9.10-cp312-cp312-win_amd64.whl.metadata (5.0 kB)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\users\anuradha\anacond
a3\lib\site-packages (from sqlalchemy) (4.11.0)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\anuradha\anaconda3\lib\s
ite-packages (from sqlalchemy) (3.0.1)
Downloading psycopg2_binary-2.9.10-cp312-cp312-win_amd64.whl (1.2 MB)
   ---------------------------------------- 0.0/1.2 MB ? eta -:--:--
   --------- ------------------------------ 0.3/1.2 MB ? eta -:--:--
   ---------------------------------------- 1.2/1.2 MB 4.8 MB/s eta 0:00:00
Installing collected packages: psycopg2-binary
Successfully installed psycopg2-binary-2.9.10
```

In [40]:
```
import pandas as pd

# Fix the path format
forest_file_path = r"C:\Users\Anuradha\Downloads\Bird_Monitoring_Data_FOREST.XLSX"
grassland_file_path = r"C:\Users\Anuradha\Downloads\Bird_Monitoring_Data_GRASSLAND.

# Load ExcelFile objects
forest_file = pd.ExcelFile(forest_file_path)
grassland_file = pd.ExcelFile(grassland_file_path)

# Define the function
def load_and_combine(excel_file, habitat_label):
    sheet_names = excel_file.sheet_names
    sheets_dict = {sheet: excel_file.parse(sheet) for sheet in sheet_names}

    combined = pd.concat(
        [df.assign(admin_unit_code=sheet_name, habitat_type=habitat_label)
         for sheet_name, df in sheets_dict.items()],
        ignore_index=True
    )
    return combined

# Load and combine sheets
forest_df = load_and_combine(forest_file, 'Forest')
grassland_df = load_and_combine(grassland_file, 'Grassland')

# Merge datasets
bird_df = pd.concat([forest_df, grassland_df], ignore_index=True)

# Check shape and preview
print(bird_df.shape)
bird_df.head()
```

(17077, 33)

Out[40]:

	Admin_Unit_Code	Sub_Unit_Code	Site_Name	Plot_Name	Location_Type	Year	Date
0	ANTI	NaN	ANTI 1	ANTI-0036	Forest	2018	2018-05-22
1	ANTI	NaN	ANTI 1	ANTI-0036	Forest	2018	2018-05-22
2	ANTI	NaN	ANTI 1	ANTI-0036	Forest	2018	2018-05-22
3	ANTI	NaN	ANTI 1	ANTI-0036	Forest	2018	2018-05-22
4	ANTI	NaN	ANTI 1	ANTI-0036	Forest	2018	2018-05-22

5 rows × 33 columns

In [42]:
```
#Standardize Column Names
bird_df.columns = bird_df.columns.str.strip().str.lower().str.replace(" ", "_")
```

In [43]:
```
#Handle Missing Values
# Summary of nulls
null_summary = bird_df.isnull().sum().sort_values(ascending=False)
print("Columns with null values:\n", null_summary[null_summary > 0])

# Optional: Drop rows with too many missing values
bird_df.dropna(thresh=10, inplace=True)  # keep rows with at least 10 non-null colu
```

```
Columns with null values:
 sub_unit_code     16355
taxoncode          8548
previously_obs     8546
npstaxoncode       8531
site_name          8531
sex                5183
distance           1486
acceptedtsn          33
id_method             2
dtype: int64
```

In [44]:
```python
#Drop Duplicates
print("Before:", bird_df.shape)
bird_df.drop_duplicates(inplace=True)
bird_df.reset_index(drop=True, inplace=True)
print("After dropping duplicates:", bird_df.shape)
```

```
Before: (17077, 33)
After dropping duplicates: (15372, 33)
```

In [45]:
```python
#Fix Data Types
# Convert to date/time and numeric types
bird_df['date'] = pd.to_datetime(bird_df['date'], errors='coerce')

numeric_cols = ['year', 'temperature', 'humidity', 'initial_three_min_cnt']
for col in numeric_cols:
    if col in bird_df.columns:
        bird_df[col] = pd.to_numeric(bird_df[col], errors='coerce')

# Convert booleans
bool_cols = ['flyover_observed', 'pif_watchlist_status', 'regional_stewardship_stat
for col in bool_cols:
    if col in bird_df.columns:
        bird_df[col] = bird_df[col].astype(str).str.upper().replace({'TRUE': True,
```

In [51]:
```python
#Add Extra Columns (Month & Season)
bird_df['month'] = bird_df['date'].dt.month
bird_df['season'] = bird_df['month'].map({
    12: 'Winter', 1: 'Winter', 2: 'Winter',
    3: 'Spring', 4: 'Spring', 5: 'Spring',
    6: 'Summer', 7: 'Summer', 8: 'Summer',
    9: 'Fall', 10: 'Fall', 11: 'Fall'
})
```

In [116…
```
from sqlalchemy import create_engine

# Replace with your PostgreSQL credentials
db_user = 'postgres'
db_password = 'anuradha24'
db_host = 'localhost'  # or '127.0.0.1'
db_port = '5432'       # default PostgreSQL port
db_name = 'bird_data'

# Create connection engine
engine = create_engine(f'postgresql+psycopg2://{db_user}:{db_password}@{db_host}:{d
```

In [118…
```
# List all column names and check for duplicates
from collections import Counter

columns = bird_df.columns
duplicates = [col for col, count in Counter(columns).items() if count > 1]
print("Duplicate columns:", duplicates)
```

```
Duplicate columns: []
```

In [120…
```
def deduplicate_columns(columns):
    seen = {}
    new_cols = []
    for col in columns:
        if col not in seen:
            seen[col] = 1
            new_cols.append(col)
        else:
            seen[col] += 1
            new_cols.append(f"{col}_{seen[col]}")
    return new_cols

bird_df.columns = deduplicate_columns(bird_df.columns)
```

In [122…
```
bird_df.to_sql('bird_monitoring', con=engine, if_exists='replace', index=False)
```

Out[122… 844

In [80]:
```
# Show null counts sorted
null_counts = bird_df.isnull().sum().sort_values(ascending=False)
print(null_counts[null_counts > 0])
```

```
sub_unit_code     14650
taxoncode          8548
previously_obs     8546
site_name          6826
npstaxoncode       6826
sex                5183
distance            689
acceptedtsn          28
id_method             2
dtype: int64
```

In [84]:
```
# First, create a mapping dictionary
distance_map = {
    '<= 50 Meters': 25,
    '50 - 100 Meters': 75,
    '100 - 200 Meters': 150,
    '200 - 500 Meters': 350,
    '> 500 Meters': 600
}

# Apply the mapping
bird_df['distance_numeric'] = bird_df['distance'].map(distance_map)

# Now you can fill NaNs with the median
bird_df['distance_numeric'] = bird_df['distance_numeric'].fillna(bird_df['distance_
```

In [86]:
```
#Fill previously_obs with "Unknown" (it's string/categorical):
bird_df['previously_obs'] = bird_df['previously_obs'].fillna('Unknown')
```

In [90]:
```
#Fill sex with "Unknown"
bird_df['sex'] = bird_df['sex'].fillna('Unknown')
```

In [92]:
```
#Fill acceptedtsn with mode (most frequent value)
bird_df['acceptedtsn'] = bird_df['acceptedtsn'].fillna(bird_df['acceptedtsn'].mode(
```

In [94]:
```
#Fill id_method with mode:
bird_df['id_method'] = bird_df['id_method'].fillna(bird_df['id_method'].mode()[0])
```

In [98]:
```
print(bird_df.columns.tolist())
```

```
['admin_unit_code', 'site_name', 'plot_name', 'location_type', 'year', 'date', 'star
t_time', 'end_time', 'observer', 'visit', 'interval_length', 'id_method', 'distanc
e', 'flyover_observed', 'sex', 'common_name', 'scientific_name', 'acceptedtsn', 'nps
taxoncode', 'aou_code', 'pif_watchlist_status', 'regional_stewardship_status', 'temp
erature', 'humidity', 'sky', 'wind', 'disturbance', 'initial_three_min_cnt', 'admin_
unit_code_2', 'habitat_type', 'taxoncode', 'previously_obs', 'month', 'season', 'tax
oncode_missing', 'distance_numeric']
```

In [100…
```
bird_df['sex_missing'] = bird_df['sex'].isnull().astype(int)
bird_df['taxoncode_missing'] = bird_df['taxoncode'].isnull().astype(int)
bird_df['previously_obs_missing'] = bird_df['previously_obs'].isnull().astype(int)
```

In [114…
```
bird_df.isnull().sum()
```

Out[114…

```
admin_unit_code                0
site_name                      0
plot_name                      0
location_type                  0
year                           0
date                           0
start_time                     0
end_time                       0
observer                       0
visit                          0
interval_length                0
id_method                      0
flyover_observed               0
sex                            0
common_name                    0
scientific_name                0
acceptedtsn                    0
aou_code                       0
pif_watchlist_status           0
regional_stewardship_status    0
temperature                    0
humidity                       0
sky                            0
wind                           0
disturbance                    0
initial_three_min_cnt          0
admin_unit_code_2              0
habitat_type                   0
taxoncode                      0
previously_obs                 0
month                          0
season                         0
taxoncode_missing              0
distance_numeric               0
sex_missing                    0
previously_obs_missing         0
dtype: int64
```

In [104…

```
bird_df.drop(columns=['distance'], inplace=True)
```

In [106…

```
bird_df.drop(columns=['npstaxoncode'], inplace=True)
```

In [110…

```
bird_df['taxoncode'] = bird_df['taxoncode'].fillna('Unknown')
```

In [124…

```
print(bird_df.shape)
```

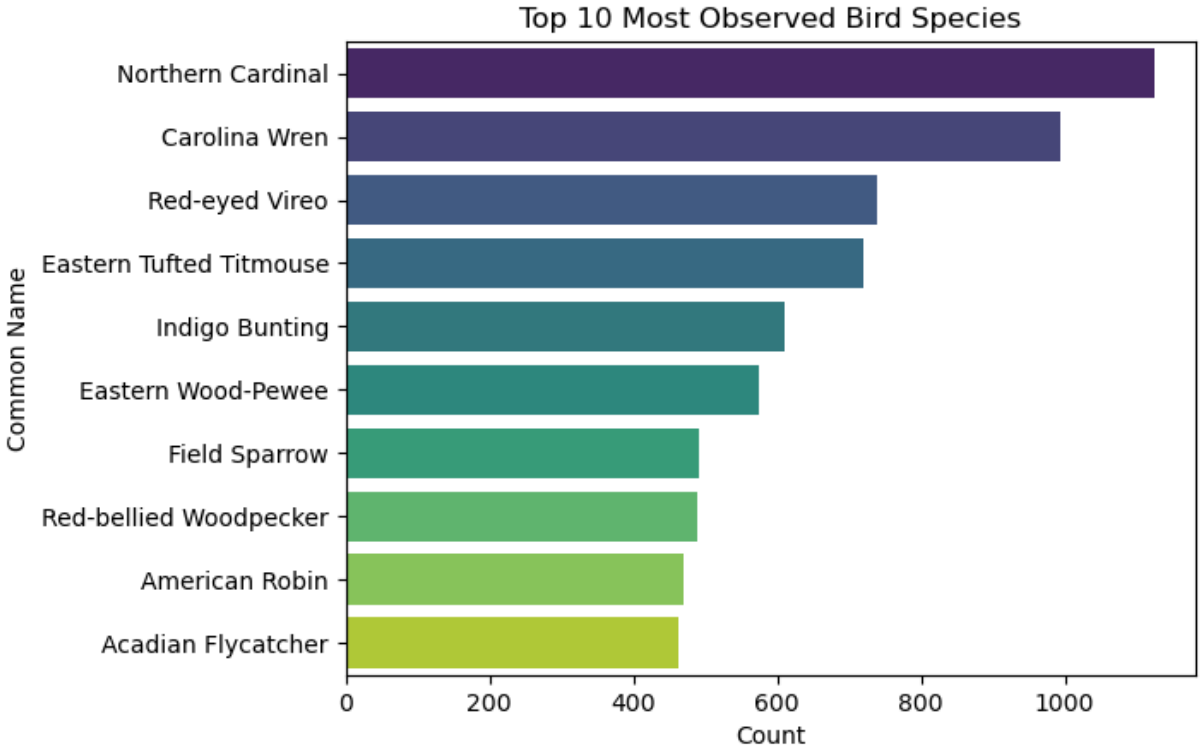
```
(15372, 36)
```

GRAPHS

In [126…

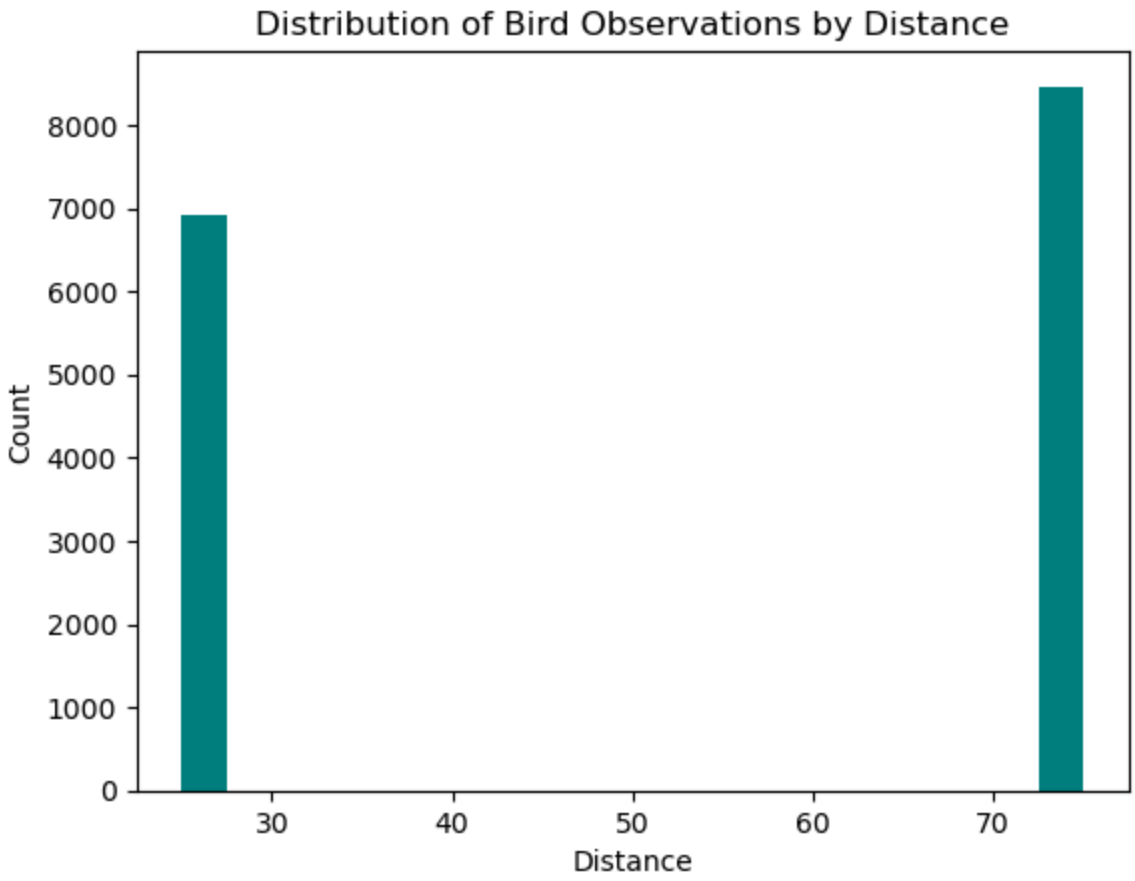
```
import seaborn as sns
import matplotlib.pyplot as plt

top_species = bird_df['common_name'].value_counts().nlargest(10)
sns.barplot(x=top_species.values, y=top_species.index, palette='viridis')
```

```
plt.title('Top 10 Most Observed Bird Species')
plt.xlabel('Count')
plt.ylabel('Common Name')
plt.show()
```



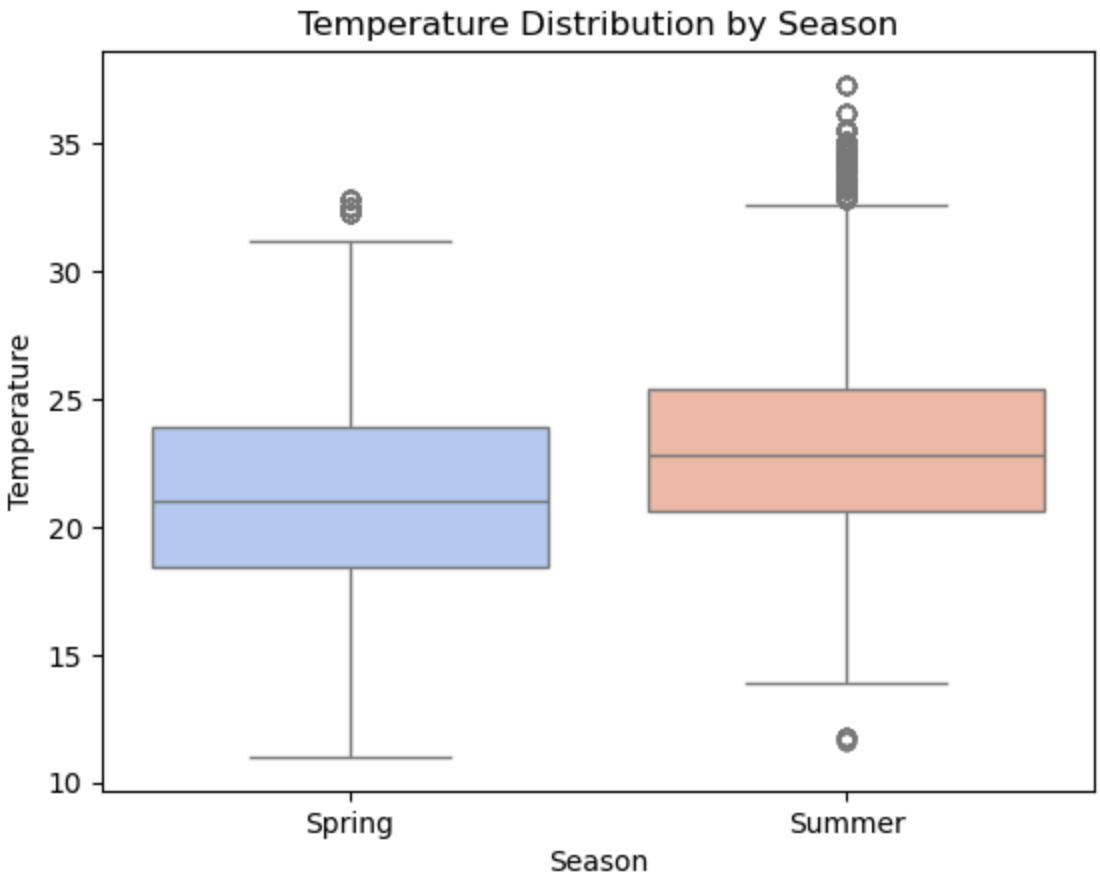
In [128…

```
plt.hist(bird_df['distance_numeric'], bins=20, color='teal')
plt.title('Distribution of Bird Observations by Distance')
plt.xlabel('Distance')
plt.ylabel('Count')
plt.show()
```



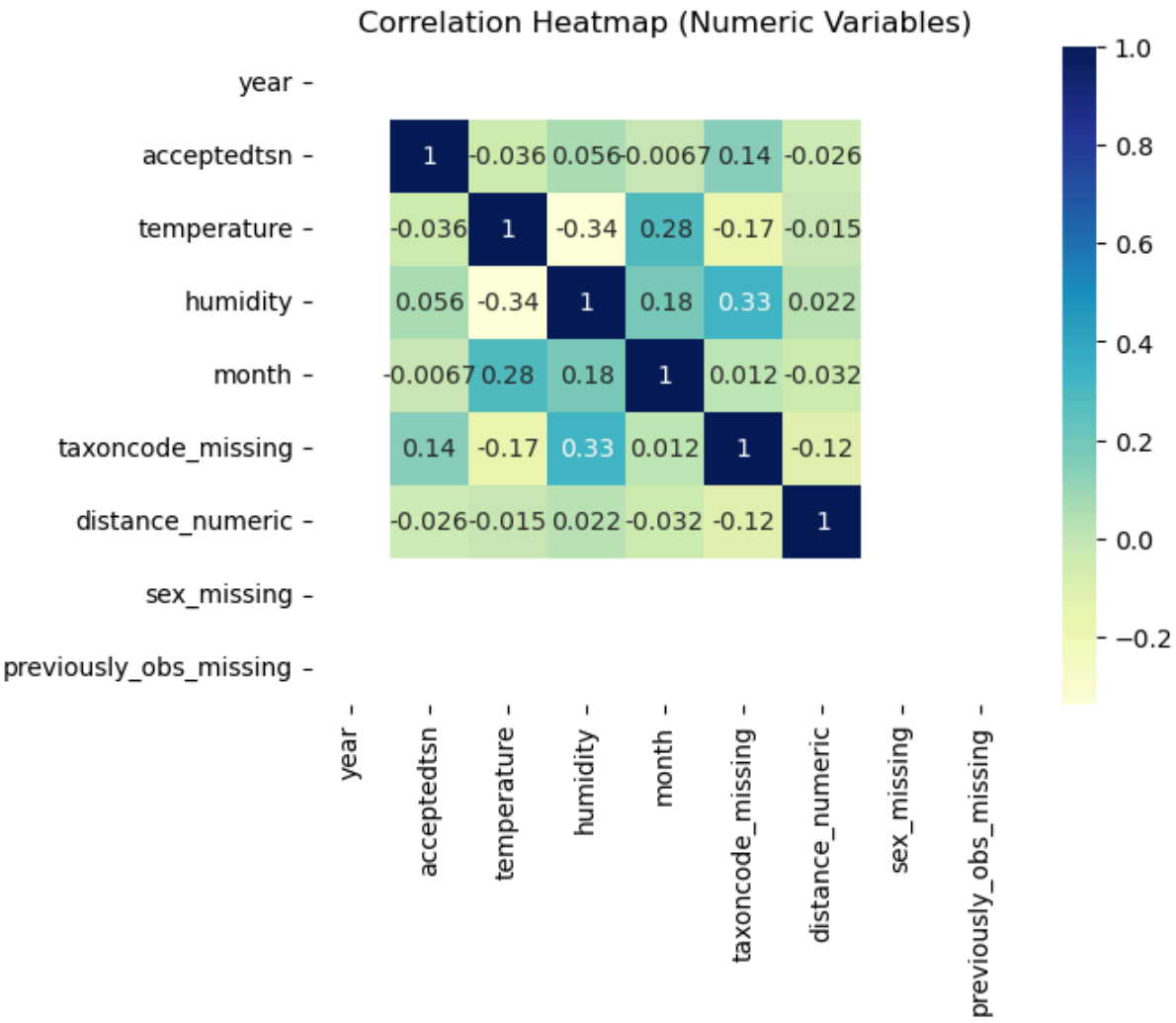
In [132…

```
sns.boxplot(x='season', y='temperature', data=bird_df, palette='coolwarm')
plt.title('Temperature Distribution by Season')
plt.xlabel('Season')
plt.ylabel('Temperature')
plt.show()
```



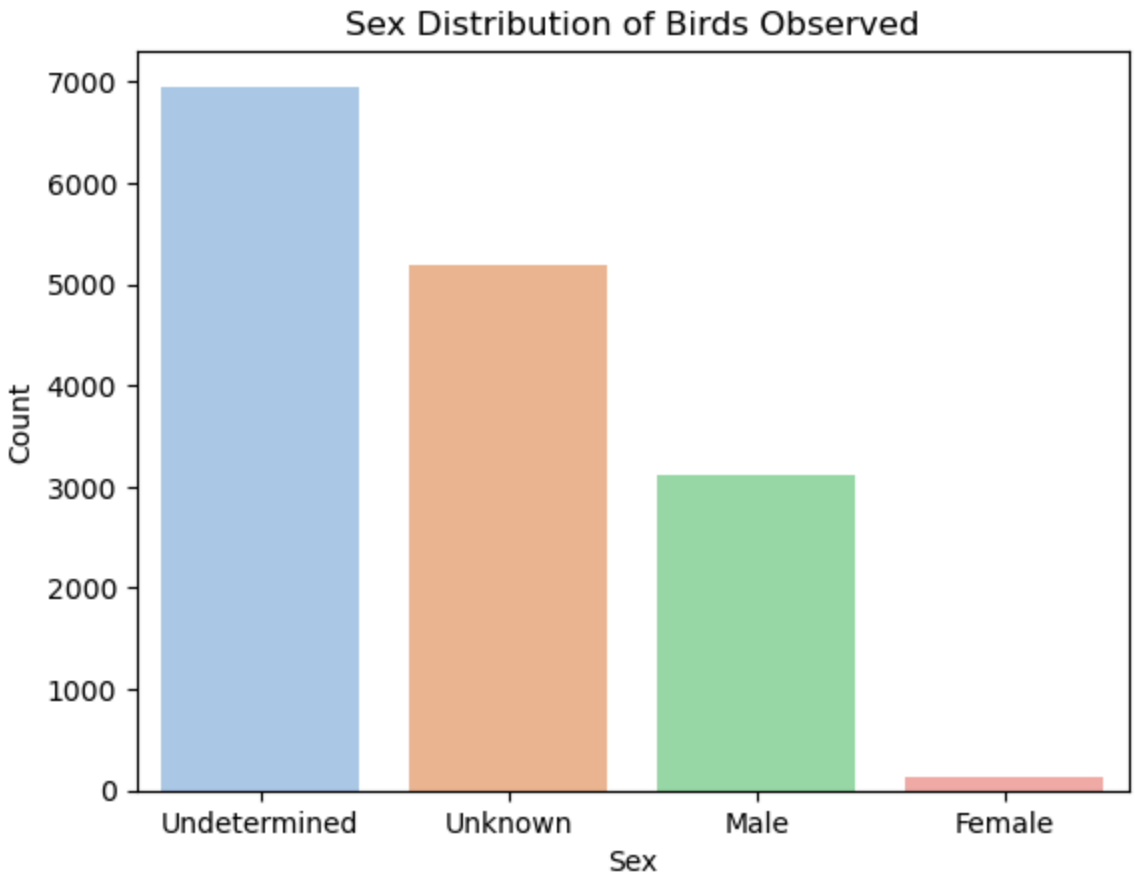
In [134…

```
import seaborn as sns

numeric_cols = bird_df.select_dtypes(include=['int', 'float']).corr()
sns.heatmap(numeric_cols, annot=True, cmap='YlGnBu')
plt.title('Correlation Heatmap (Numeric Variables)')
plt.show()
```



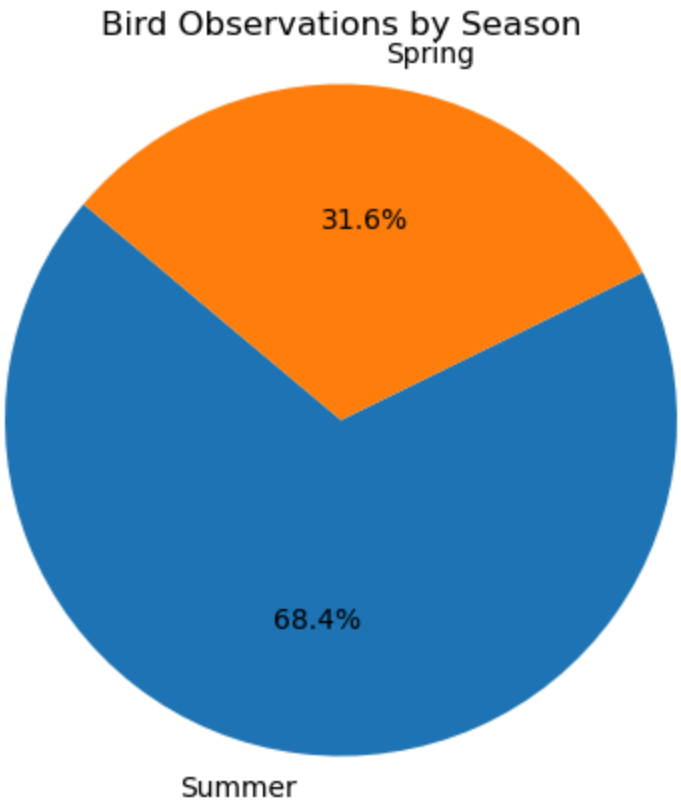
In [136…

```
sns.countplot(x='sex', data=bird_df, palette='pastel')
plt.title('Sex Distribution of Birds Observed')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.show()
```



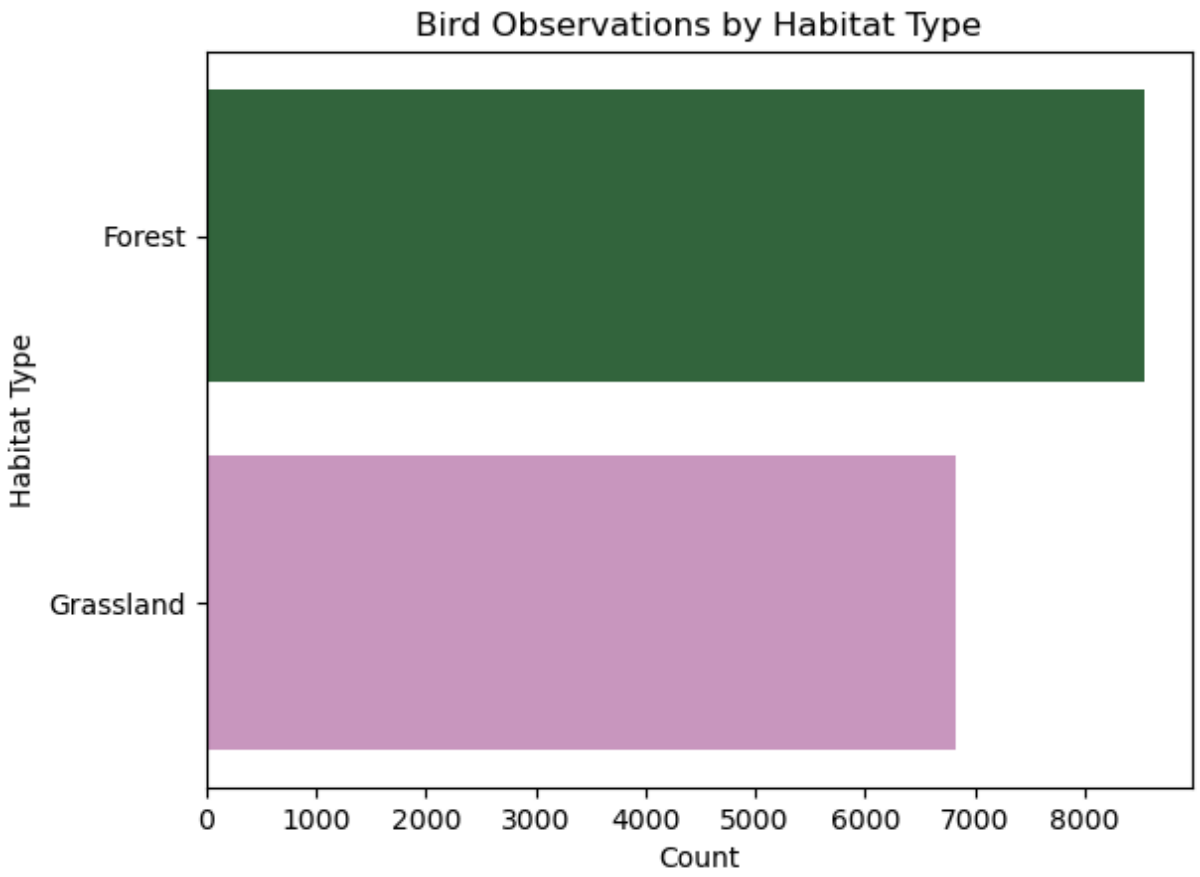
In [138…

```
season_counts = bird_df['season'].value_counts()
plt.pie(season_counts, labels=season_counts.index, autopct='%1.1f%%', startangle=14
plt.title('Bird Observations by Season')
plt.axis('equal')
plt.show()
```



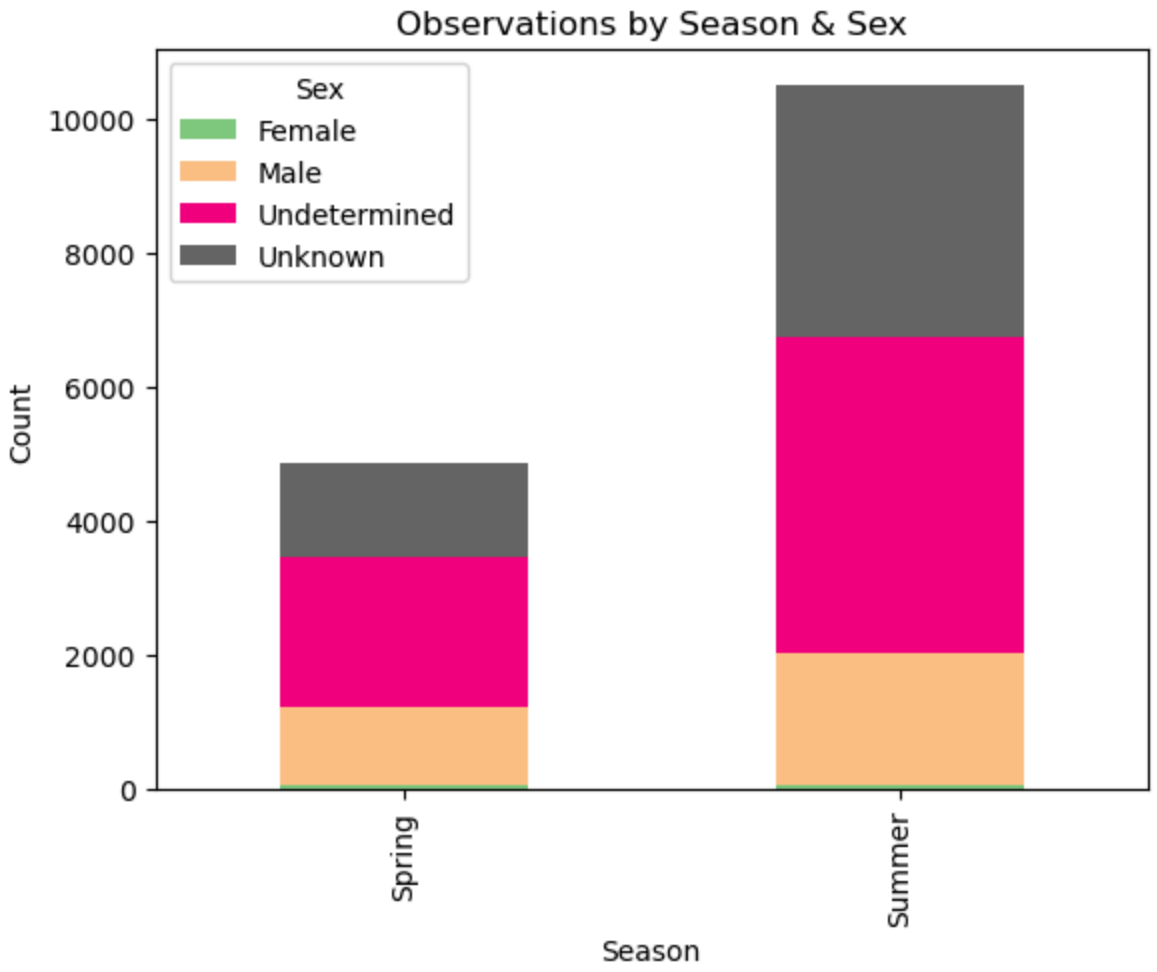
In [140…

```
sns.countplot(y='habitat_type', data=bird_df, order=bird_df['habitat_type'].value_c
plt.title('Bird Observations by Habitat Type')
plt.xlabel('Count')
plt.ylabel('Habitat Type')
plt.show()
```



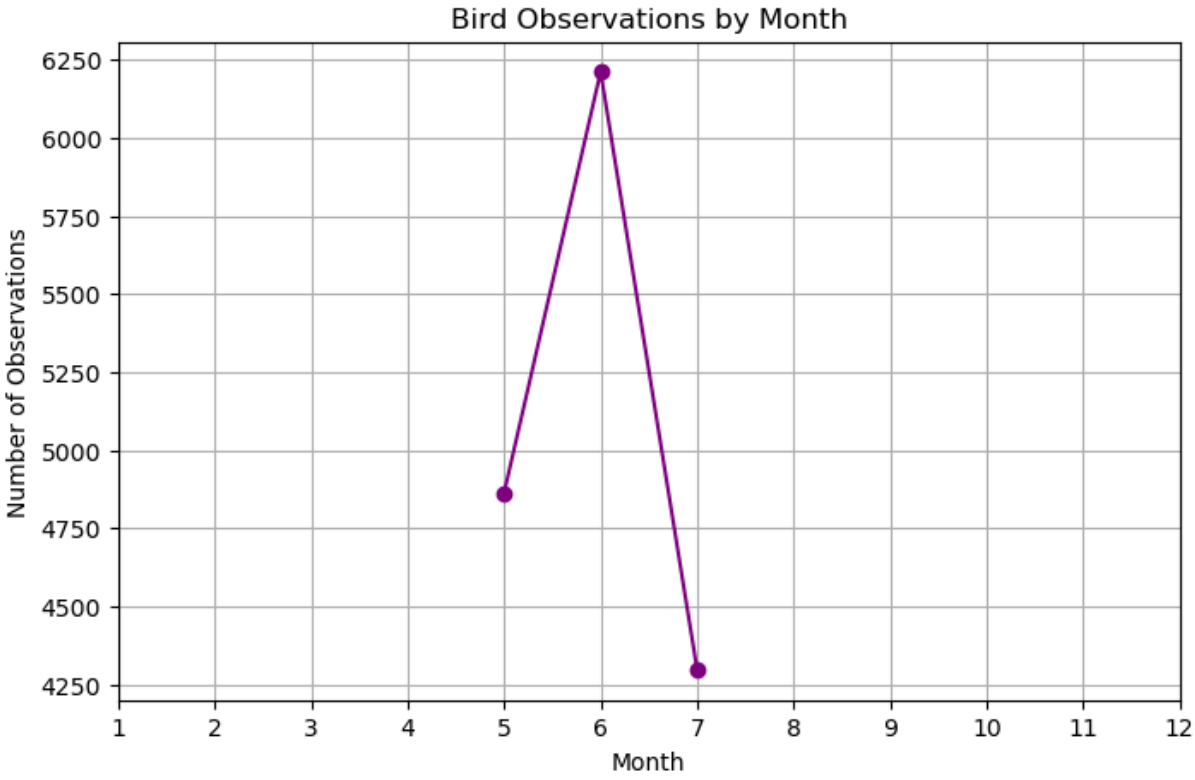
In [146…

```
season_sex_counts = bird_df.groupby(['season', 'sex']).size().unstack(fill_value=0)
season_sex_counts.plot(kind='bar', stacked=True, colormap='Accent')
plt.title('Observations by Season & Sex')
plt.xlabel('Season')
plt.ylabel('Count')
plt.legend(title='Sex')
plt.show()
```



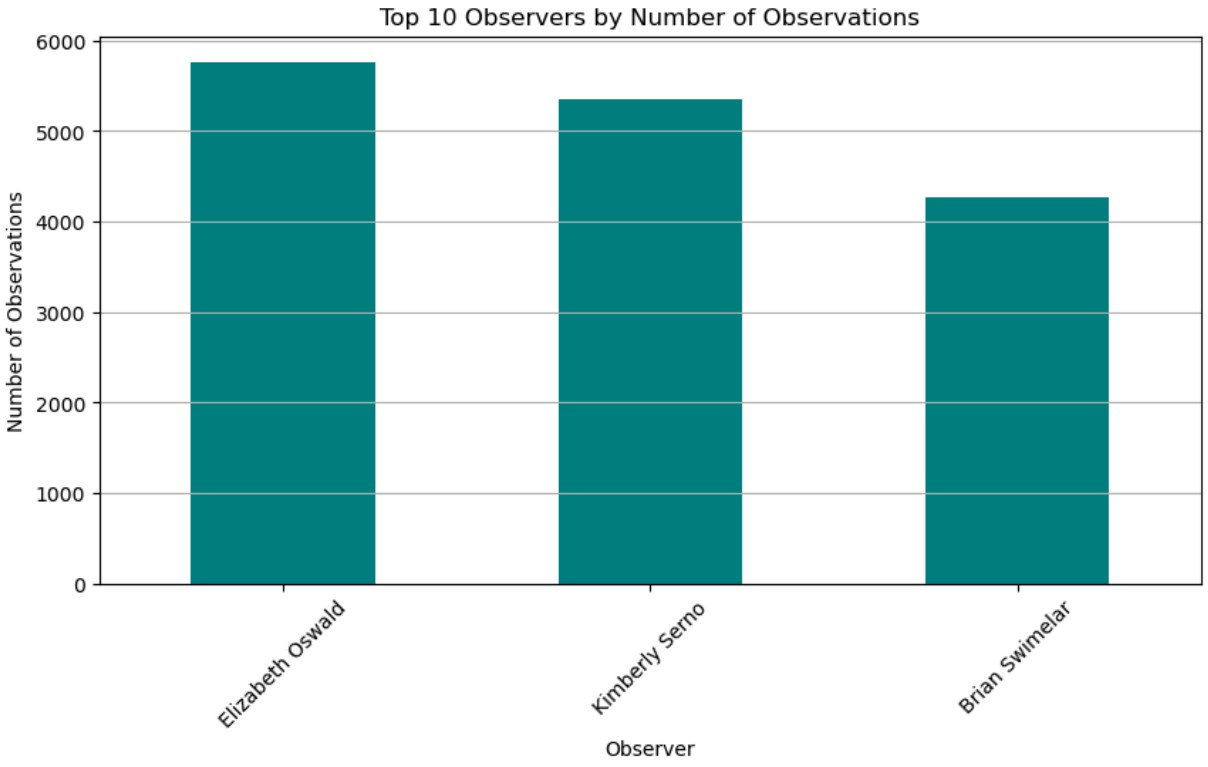
In [150…

```
monthly_counts = bird_df['month'].value_counts().sort_index()

plt.figure(figsize=(8, 5))
monthly_counts.plot(kind='line', marker='o', color='purple')
plt.title('Bird Observations by Month')
plt.xlabel('Month')
plt.ylabel('Number of Observations')
plt.grid(True)
plt.xticks(range(1, 13))
plt.show()
```



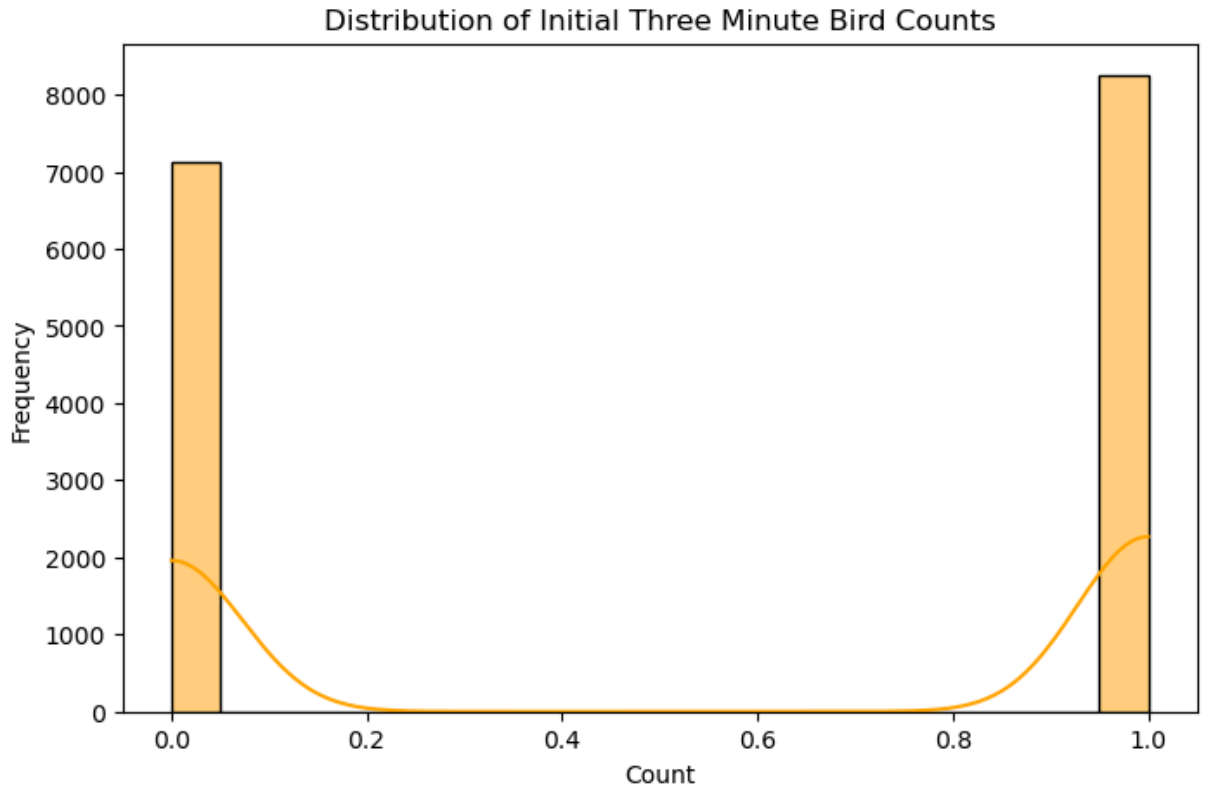
In [152…

```
top_observers = bird_df['observer'].value_counts().head(10)

plt.figure(figsize=(10, 5))
top_observers.plot(kind='bar', color='teal')
plt.title('Top 10 Observers by Number of Observations')
plt.xlabel('Observer')
plt.ylabel('Number of Observations')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```



In [156…

```
plt.figure(figsize=(8,5))
sns.histplot(data=bird_df, x='initial_three_min_cnt', bins=20, kde=True, color='ora
plt.title('Distribution of Initial Three Minute Bird Counts')
plt.xlabel('Count')
plt.ylabel('Frequency')
plt.show()
```

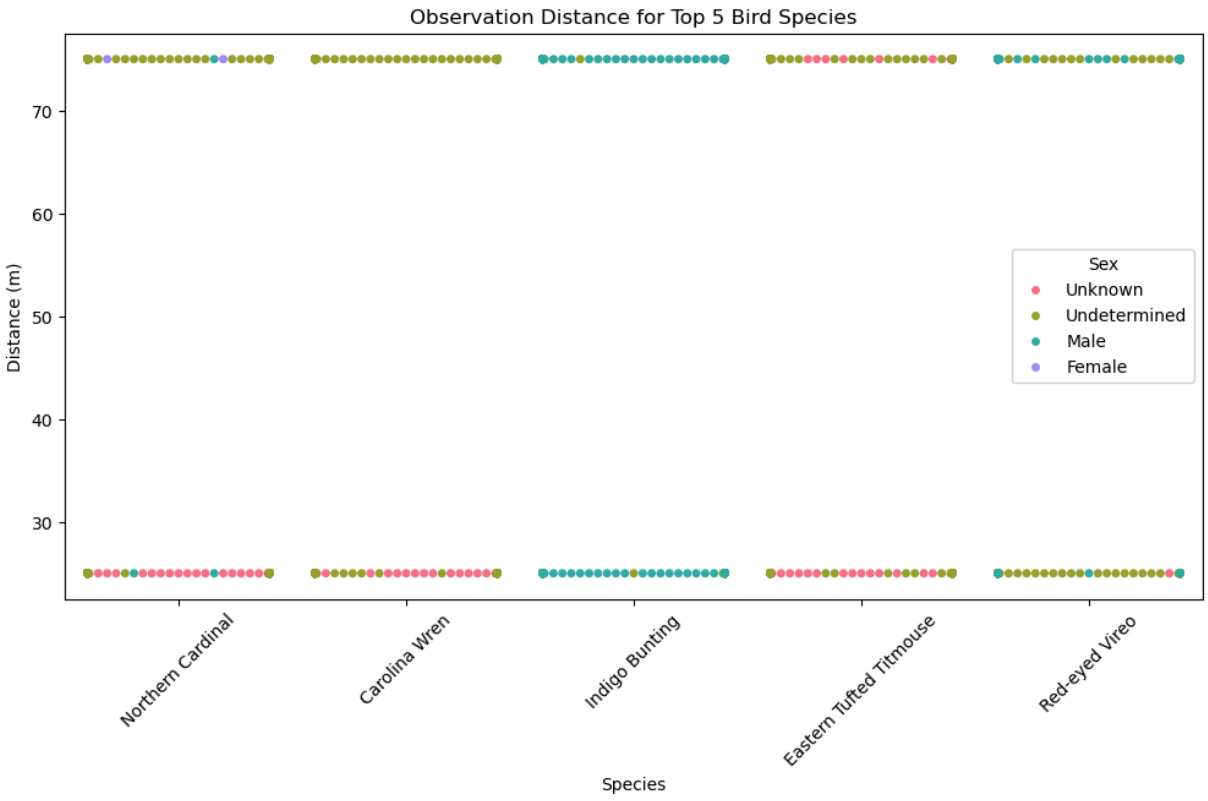


In [160…

```
plt.figure(figsize=(8,6))
sns.violinplot(data=bird_df, x='sex', y='distance_numeric', palette='Set3')
plt.title('Observation Distance by Sex')
plt.xlabel('Sex')
plt.ylabel('Distance (m)')
plt.show()
```

In [174…

```
top_species = bird_df['common_name'].value_counts().head(5).index
filtered = bird_df[bird_df['common_name'].isin(top_species)]

plt.figure(figsize=(12,6))
sns.swarmplot(data=filtered, x='common_name', y='distance_numeric', hue='sex', pale
plt.title('Observation Distance for Top 5 Bird Species')
plt.xlabel('Species')
plt.ylabel('Distance (m)')
plt.legend(title='Sex')
plt.xticks(rotation=45)
plt.show()
```



In [167…
```
pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\anuradha\anaconda3\lib\site-packag
es (5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\anuradha\anaconda3\lib\si
te-packages (from plotly) (8.2.3)
Requirement already satisfied: packaging in c:\users\anuradha\anaconda3\lib\site-pac
kages (from plotly) (24.1)
Note: you may need to restart the kernel to use updated packages.
```

In [168…
```
import plotly.express as px

top_species = bird_df['common_name'].value_counts().head(10).reset_index()
top_species.columns = ['Species', 'Count']

fig = px.bar(top_species, x='Species', y='Count',
             title='Top 10 Most Observed Bird Species',
             color='Count', color_continuous_scale='Viridis',
             text='Count')

fig.update_traces(textposition='outside')
fig.update_layout(xaxis_tickangle=-45)
fig.show()
```

In [169…

```
monthly_data = bird_df.groupby('month').size().reset_index(name='Observations')

fig = px.line(monthly_data, x='month', y='Observations',
              markers=True, title='Monthly Observation Trend')

fig.update_layout(xaxis_title='Month', yaxis_title='Number of Observations')
fig.show()
```

In [175…

```
fig = px.scatter(bird_df, x='distance_numeric', y='initial_three_min_cnt',
                 color='habitat_type',
                 hover_data=['common_name', 'location_type', 'observer'],
                 title='Distance vs Bird Count by Habitat Type')

fig.update_layout(xaxis_title='Distance (m)', yaxis_title='Bird Count')
fig.show()
```

In [176…

```
fig = px.sunburst(bird_df,
                  path=['season', 'habitat_type', 'common_name'],
                  values=None,
                  title='Bird Observations by Season → Habitat → Species')

fig.show()
```

In [178…

```
pip install dash
```

```
Collecting dash
  Downloading dash-3.2.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: Flask<3.2,>=1.0.4 in c:\users\anuradha\anaconda3\lib
\site-packages (from dash) (3.0.3)
Requirement already satisfied: Werkzeug<3.2 in c:\users\anuradha\anaconda3\lib\site-
packages (from dash) (3.0.3)
Requirement already satisfied: plotly>=5.0.0 in c:\users\anuradha\anaconda3\lib\site
-packages (from dash) (5.24.1)
Requirement already satisfied: importlib-metadata in c:\users\anuradha\anaconda3\lib
\site-packages (from dash) (7.0.1)
Requirement already satisfied: typing-extensions>=4.1.1 in c:\users\anuradha\anacond
a3\lib\site-packages (from dash) (4.11.0)
Requirement already satisfied: requests in c:\users\anuradha\anaconda3\lib\site-pack
ages (from dash) (2.32.3)
Collecting retrying (from dash)
  Downloading retrying-1.4.2-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: nest-asyncio in c:\users\anuradha\anaconda3\lib\site-
packages (from dash) (1.6.0)
Requirement already satisfied: setuptools in c:\users\anuradha\anaconda3\lib\site-pa
ckages (from dash) (75.1.0)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\anuradha\anaconda3\lib\site
-packages (from Flask<3.2,>=1.0.4->dash) (3.1.4)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\anuradha\anaconda3\li
b\site-packages (from Flask<3.2,>=1.0.4->dash) (2.2.0)
Requirement already satisfied: click>=8.1.3 in c:\users\anuradha\anaconda3\lib\site-
packages (from Flask<3.2,>=1.0.4->dash) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\anuradha\anaconda3\lib\sit
e-packages (from Flask<3.2,>=1.0.4->dash) (1.6.2)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\anuradha\anaconda3\lib\si
te-packages (from plotly>=5.0.0->dash) (8.2.3)
Requirement already satisfied: packaging in c:\users\anuradha\anaconda3\lib\site-pac
kages (from plotly>=5.0.0->dash) (24.1)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\anuradha\anaconda3\lib
\site-packages (from Werkzeug<3.2->dash) (2.1.3)
Requirement already satisfied: zipp>=0.5 in c:\users\anuradha\anaconda3\lib\site-pac
kages (from importlib-metadata->dash) (3.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\anuradha\anacond
a3\lib\site-packages (from requests->dash) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\anuradha\anaconda3\lib\site-
packages (from requests->dash) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\anuradha\anaconda3\lib
\site-packages (from requests->dash) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\anuradha\anaconda3\lib
\site-packages (from requests->dash) (2025.1.31)
Requirement already satisfied: colorama in c:\users\anuradha\anaconda3\lib\site-pack
ages (from click>=8.1.3->Flask<3.2,>=1.0.4->dash) (0.4.6)
Downloading dash-3.2.0-py3-none-any.whl (7.9 MB)
   ---------------------------------------- 0.0/7.9 MB ? eta -:--:--
   -- ------------------------------------- 0.5/7.9 MB 4.2 MB/s eta 0:00:02
   ------- -------------------------------- 1.6/7.9 MB 5.2 MB/s eta 0:00:02
   ------------- -------------------------- 2.6/7.9 MB 5.0 MB/s eta 0:00:02
   -------------- ------------------------- 3.1/7.9 MB 4.4 MB/s eta 0:00:02
   ------------------ --------------------- 3.7/7.9 MB 3.9 MB/s eta 0:00:02
   ----------------------- ---------------- 4.7/7.9 MB 4.0 MB/s eta 0:00:01
   ------------------------------ --------- 6.0/7.9 MB 4.4 MB/s eta 0:00:01
   ---------------------------------- ----- 7.1/7.9 MB 4.5 MB/s eta 0:00:01
```

```
   ---------------------------------------- 7.9/7.9 MB 4.5 MB/s eta 0:00:00
Downloading retrying-1.4.2-py3-none-any.whl (10 kB)
Installing collected packages: retrying, dash
Successfully installed dash-3.2.0 retrying-1.4.2
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```
python dash_app.py
```

In [186…
```
df.to_excel("cleaned_data.xlsx", index=False)
```

In [188…
```
bird_df.to_excel(r'C:\Users\Anuradha\Documents\bird_data.xlsx', index=False)
```

In [194…
```
query = "SELECT * FROM bird_monitoring"
df = pd.read_sql(query, con=engine)
```

In [192…
```
# Replace with your actual connection details
engine = create_engine('postgresql://postgres:anuradha24@localhost:5432/bird_data')
```

In [196…
```
import pandas as pd
from sqlalchemy import create_engine
import os

# ✅ Update these details with your actual credentials
engine = create_engine('postgresql://postgres:anuradha24@localhost:5432/bird_data')

# Run the SELECT query
query = "SELECT * FROM bird_monitoring"
df = pd.read_sql(query, con=engine)

# Save to Excel in your Documents folder
file_path = os.path.expanduser("~/Documents/bird_monitoring_export.xlsx")
df.to_excel(file_path, index=False)

print(f"✔ Exported to: {file_path}")
```

```
✔ Exported to: C:\Users\Anuradha/Documents/bird_monitoring_export.xlsx
```

In [208…
```
!pip install pyppeteer
jupyter nbconvert --to webpdf Bird_data_analysis.ipynb
```

```
  Cell In[208], line 2
    jupyter nbconvert --to webpdf Bird_data_analysis.ipynb
            ^
SyntaxError: invalid syntax
```

In [210…
```
!jupyter nbconvert --to webpdf Bird_data_analysis.ipynb
```

```
[NbConvertApp] Converting notebook Bird_data_analysis.ipynb to webpdf
[NbConvertApp] WARNING | Alternative text is missing on 13 image(s).
[NbConvertApp] Building PDF
Traceback (most recent call last):
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\exporters\webpdf.p
y", line 78, in main
    from playwright.async_api import async_playwright  # type: ignore[import-not-fou
nd]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ModuleNotFoundError: No module named 'playwright'

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "C:\Users\Anuradha\anaconda3\Scripts\jupyter-nbconvert-script.py", line 10, i
n <module>
    sys.exit(main())
             ^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\jupyter_core\application.py",
line 283, in launch_instance
    super().launch_instance(argv=argv, **kwargs)
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\traitlets\config\application.p
y", line 1075, in launch_instance
    app.start()
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\nbconvertapp.py", li
ne 420, in start
    self.convert_notebooks()
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\nbconvertapp.py", li
ne 597, in convert_notebooks
    self.convert_single_notebook(notebook_filename)
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\nbconvertapp.py", li
ne 563, in convert_single_notebook
    output, resources = self.export_single_notebook(
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\nbconvertapp.py", li
ne 487, in export_single_notebook
    output, resources = self.exporter.from_filename(
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\exporters\templateex
porter.py", line 386, in from_filename
    return super().from_filename(filename, resources, **kw)  # type:ignore[return-va
lue]
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\exporters\exporter.p
y", line 201, in from_filename
    return self.from_file(f, resources=resources, **kw)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\exporters\templateex
porter.py", line 392, in from_file
    return super().from_file(file_stream, resources, **kw)  # type:ignore[return-val
ue]
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\exporters\exporter.p
y", line 220, in from_file
    return self.from_notebook_node(
           ^^^^^^^^^^^^^^^^^^^^^^^
```

```
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\exporters\webpdf.p
y", line 174, in from_notebook_node
    pdf_data = self.run_playwright(html)
               ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\exporters\webpdf.p
y", line 163, in run_playwright
    pdf_data = pool.submit(run_coroutine, main(temp_file)).result()
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\concurrent\futures\_base.py", line 456, in r
esult
    return self.__get_result()
           ^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\concurrent\futures\_base.py", line 401, in _
_get_result
    raise self._exception
  File "C:\Users\Anuradha\anaconda3\Lib\concurrent\futures\thread.py", line 58, in r
un
    result = self.fn(*self.args, **self.kwargs)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\exporters\webpdf.p
y", line 161, in run_coroutine
    return loop.run_until_complete(coro)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\asyncio\base_events.py", line 687, in run_un
til_complete
    return future.result()
           ^^^^^^^^^^^^^^^
  File "C:\Users\Anuradha\anaconda3\Lib\site-packages\nbconvert\exporters\webpdf.p
y", line 84, in main
    raise RuntimeError(msg) from e
RuntimeError: Playwright is not installed to support Web PDF conversion. Please inst
all `nbconvert[webpdf]` to enable.
```

In [ ]: