```
import pandas as pd


# Create data
data = {
 'species': [
 'Tiger', 'Rhino', 'Elephant', 'Lion', 'Leopard', 'Wolf', # Endangered
 'Deer', 'Rabbit', 'Zebra', 'Fox', 'Squirrel', 'Cow', 'Goat', # Least Concern
 'Panda', 'Eagle', 'Penguin', 'Seal', 'Kangaroo', 'Koala', 'Owl' # Vulnerable
 ],
 'endangered_status': (
 ['Endangered'] * 6 + ['Least Concern'] * 7 + ['Vulnerable'] * 7
 )
}


df = pd.DataFrame(data)


# Group by status and count
result = df.groupby('endangered_status')['species'].count().reset_index()
result.columns = ['Endangered Status', 'Species Count']
print(result)




import pandas as pd


# Create data
data = {
```

```
 'species': [
 'Tiger', 'Rhino', 'Elephant', 'Lion', 'Leopard', 'Wolf', # Endangered (6)
 'Deer', 'Rabbit', 'Zebra', 'Fox', 'Squirrel', 'Cow', 'Goat', # Least Concern (7)
 'Panda', 'Eagle', 'Penguin', 'Seal', 'Kangaroo', 'Koala', 'Owl' # Vulnerable (7)
 ],
 'endangered_status': (
 ['Endangered'] * 6 + ['Least Concern'] * 7 + ['Vulnerable'] * 7
 )
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Group by endangered status and count species
result = df.groupby('endangered_status')['species'].count().reset_index()
result.columns = ['Endangered Status', 'Species Count']

# Display result
print(result)

 import pandas as pd
import numpy as np

# Create dataset
data = {
 'species': ['Tiger'] * 9,
 'year': list(range(2015, 2024)),
```

```
 'population': [1210, 1440, 1440, 1440, 1330, 1245, 1355, 1320, 1320]
}

# Create DataFrame
df = pd.DataFrame(data)

# Sort by year (for time sequence)
df = df.sort_values('year')

# Fill missing values if any
df['population'] = df.groupby('species')['population'].ffill().bfill()

# Calculate yearly population percentage change
df['population_change_%'] = df['population'].pct_change() * 100

# Display result
print(df)
```

```
import pandas as pd

# Example data
data = {
 'species': ['Elephant', 'Kangaroo', 'Orangutan', 'Panda', 'Rhino', 'Snow leopard', 'Tiger'],
 'population': [1220, 440, 980, 950, 1420, 630, 1250]
}
```

```
# Create DataFrame

df = pd.DataFrame(data)


# Calculate mean population per species

mean_pop = df.groupby('species')['population'].mean().reset_index()


# Display result

print(mean_pop)

import pandas as pd


# Create data

data = {

 'species': [

 'Tiger', 'Rhino', 'Elephant', 'Lion', 'Leopard', 'Wolf', # Endangered (6)

 'Deer', 'Rabbit', 'Zebra', 'Fox', 'Squirrel', 'Cow', 'Goat', # Least Concern (7)

 'Panda', 'Eagle', 'Penguin', 'Seal', 'Kangaroo', 'Koala', 'Owl' # Vulnerable (7)

 ],

 'endangered_status': (

 ['Endangered'] * 6 + ['Least Concern'] * 7 + ['Vulnerable'] * 7

 )

}


# Create DataFrame

df = pd.DataFrame(data)


# Group by endangered status and count species

result = df.groupby('endangered_status')['species'].count().reset_index()
```

```
result.columns = ['Endangered Status', 'Species Count']


# Display result

print(result)

import pandas as pd

import numpy as np


# Create dataset

data = {

 'species': ['Tiger'] * 9,

 'year': list(range(2015, 2024)),

 'population': [1210, 1440, 1440, 1440, 1330, 1245, 1355, 1320, 1320]

}


# Create DataFrame

df = pd.DataFrame(data)


# Sort by year (for proper time order)

df = df.sort_values('year')


# Fill any missing population values

df['population'] = df.groupby('species')['population'].ffill().bfill()


# Calculate yearly population percentage change

df['population_change_%'] = df['population'].pct_change() * 100


# Display result
```

```
print(df)


import pandas as pd

import numpy as np


# Create dataset

data = {

 'species': (['Rhino']*9 + ['Tiger']*9 + ['Elephant']*9 +

 ['Orangutan']*9 + ['Panda']*9),

 'year': list(range(2015, 2024)) * 5,

 'population': [

 1630, 1190, 1240, 1370, 1360, 1350, 1550, 1750, 1680, # Rhino

 1210, 1420, 1420, 1420, 1350, 1240, 1370, 1350, 1320, # Tiger

 1390, 1150, 1230, 1200, 1200, 1300, 1550, 1740, 1650, # Elephant

 800, 1150, 1150, 1000, 1060, 770, 1030, 780, 1120, # Orangutan

 890, 950, 1050, 850, 1000, 920, 820, 890, 1000 # Panda

 ]

}


# Create DataFrame

df = pd.DataFrame(data)


# Forward-fill and backward-fill missing population values for each species

df['population'] = df.groupby('species')['population'].ffill().bfill()


# Display the final DataFrame

print(df)
```

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt


# ---------- SAMPLE DATA ----------

np.random.seed(0) # For consistent results every time

data = {

 'species': [f'Sp{i}' for i in range(1, 31)],

 'status': np.random.choice(['Endangered', 'Least Concern', 'Vulnerable'], 30),

 'region': np.random.choice(['North', 'Unknown', 'South', 'East', 'West'], 30),

 'population': np.random.randint(0, 1001, 30)

}


# Create DataFrame

df = pd.DataFrame(data)


# ----------        SPECIES COUNT BY STATUS ----------

plt.figure(figsize=(6, 4))

sns.countplot(data=df, x='status')

plt.title('Species Count by Endangered Status')

plt.xlabel('Endangered Status')

plt.ylabel('Count of Species')

plt.show()


# ----------        HEATMAP: MEAN POPULATION BY REGION & STATUS ----------
```

```
heat = df.pivot_table(values='population', index='region', columns='status', aggfunc='mean')

plt.figure(figsize=(6, 4))

sns.heatmap(heat, annot=True, fmt=".0f", cmap='YlOrRd')

plt.title('Mean Population by Region and Status')

plt.xlabel('Endangered Status')

plt.ylabel('Region')

plt.show()


import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt


# ---------- SAMPLE DATA ----------

np.random.seed(0) # For consistent results every time

data = {

 'species': [f'Sp{i}' for i in range(1, 31)],

 'status': np.random.choice(['Endangered', 'Least Concern', 'Vulnerable'], 30),

 'region': np.random.choice(['North', 'Unknown', 'South', 'East', 'West'], 30),

 'population': np.random.randint(0, 1001, 30)

}


# Create DataFrame

df = pd.DataFrame(data)


# ----------        SPECIES COUNT BY STATUS ----------

plt.figure(figsize=(6, 4))
```

```
sns.countplot(data=df, x='status')

plt.title('Species Count by Endangered Status')

plt.xlabel('Endangered Status')

plt.ylabel('Count of Species')

plt.show()


# ----------          HEATMAP: MEAN POPULATION BY REGION & STATUS ----------

heat = df.pivot_table(values='population', index='region', columns='status', aggfunc='mean')

plt.figure(figsize=(6, 4))

sns.heatmap(heat, annot=True, fmt=".0f", cmap='YlOrRd')

plt.title('Mean Population by Region and Status')

plt.xlabel('Endangered Status')

plt.ylabel('Region')
```