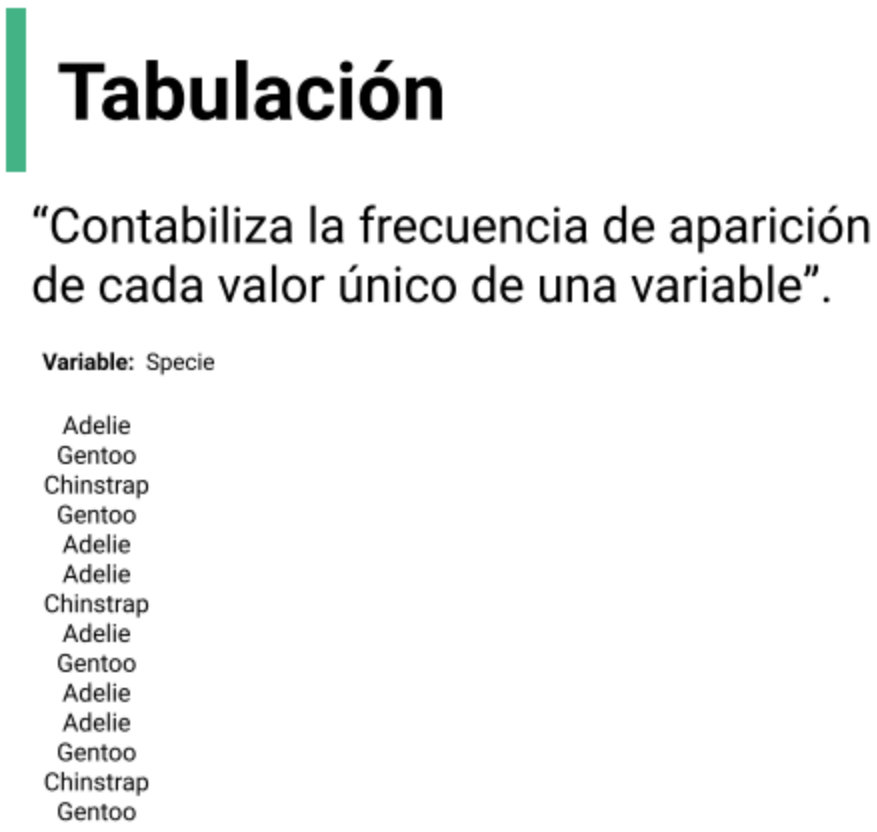
Explorando una variable categórica: conteos y proporciones

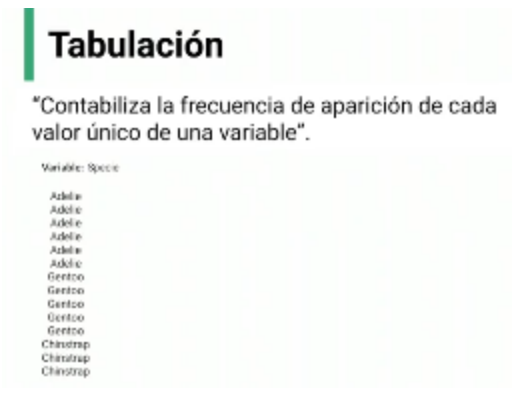
Los conteos y proporciones se basan en la tabulación.

La tabulación contabiliza la frecuencia de aparición de cada valor único de una variable.
Por ejemplo en el caso de nuestra especies de los pingüinos, tenemos 3 especies:

- Adelie:
- Gentoo
- Chinstrap

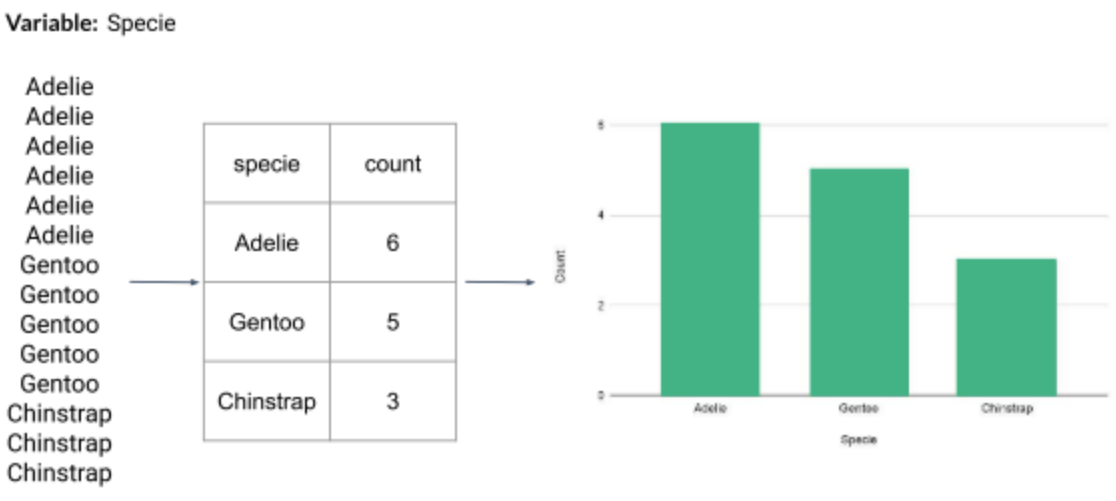


Para llevar un mejor control, podemos ordenar por orden alfabético o por cantidad de apariciones



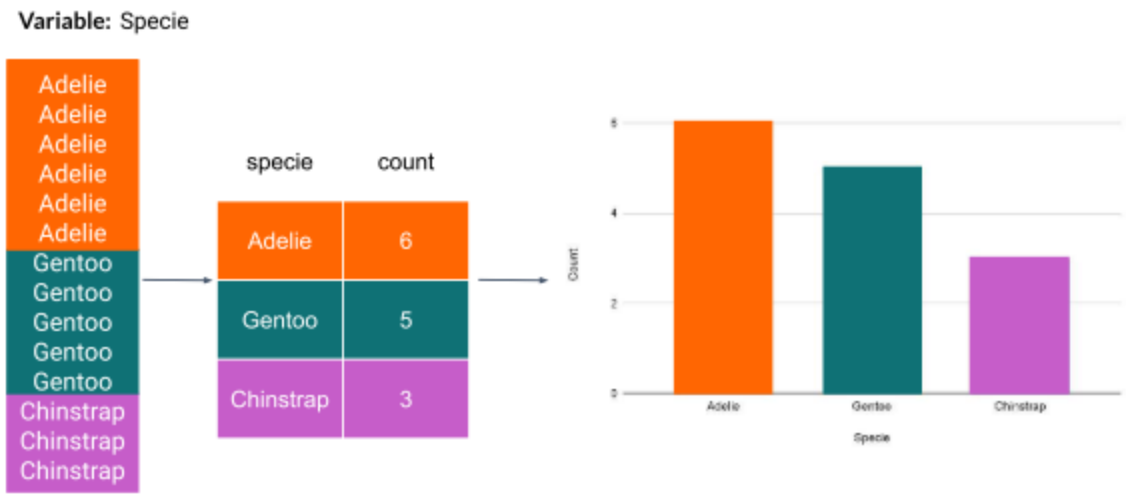
Conteos

Una vez ordenados es más contar cuantos elementos tenemos de cada elemento único. Pero la palabra tabulación viene de tabla así que pasamos esto a una tabla y decir lo mismo pero de manera resumida. Para después realizar un diagrama de frecuencia. En el cual graficamos el numero de coincidencias por cada valor de la variable.



Así nosotros de manera visual podemos obtener información de la variable, en este caso podemos ver que la especie que mas ejemplares tiene es la Adelie y la que menos tiene es la Chinstrap.

Ademas puedes llevarlo a nivel mas arriba haciendo una segmentación por color a cada especie.



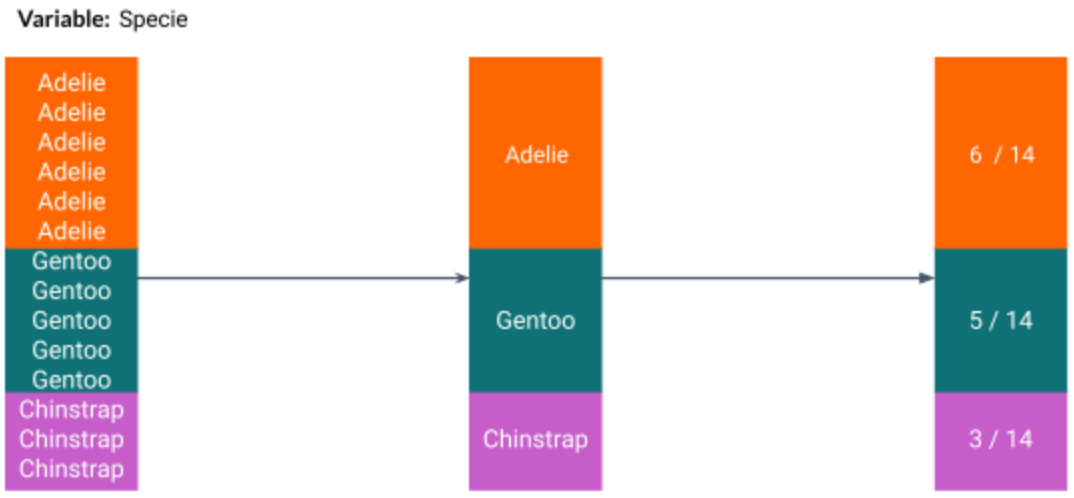
Así de esta forma el color nos ayuda aún mas a identificar a las especies. Así ya podemos asociar un color por especie en cada fase; desde conteo, tabulación, creación de gráficas, entre otros.

Este es la forma en que procedemos a realizar la etapa de conteos.

Proporciones

La proporción se refiere a la relación de correspondencia entre las partes y el todo.

En el ejemplo anterior tuvimos una cantidad de 14 pingüinos exactamente, así los podemos clasificar por proporciones, cómo sigue:



Esta división me dice la proporción que tengo con respecto de una variable y sus posibles valores. Esta manera es una forma sencilla de mostrar los conteos por proporciones.

Pasemos a realizarlo en Deepnote

In [ ]:
```
# Importando librerías
import empiricaldist
import janitor
import matplotlib.pyplot as plt
import numpy as np
import palmerpenguins
import pandas as pd
import scipy.stats
import seaborn as sns
import sklearn.metrics
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats as ss
import session_info
```

Establecer apariencia general de las gráficas

In [ ]:
```
%matplotlib inline
sns.set_style(style='whitegrid')
sns.set_context(context='notebook')
plt.rcParams['figure.figsize'] = (11, 9.4)

penguin_color = {
    'Adelie': '#ff6602ff',
    'Gentoo': '#0f7175ff',
    'Chinstrap': '#c65dc9ff'
}
```

Cargar los datos

Datos crudos

In [ ]:
```
raw_penguins_df = pd.read_csv('dataset/penguins_raw.csv')
```

Datos Preprocesados

NOTA: Puede que no usemos estos datos

In [ ]:
```
preprocessed_penguins_df = pd.read_csv('dataset/penguins.csv')
```

Conteos y proporciones

Hagamos una pregunta, ¿qué parámetros estadísticos describen el conjunto de datos en general?

Todas las variables

In [ ]:
```
preprocessed_penguins_df.describe()
```

Out[ ]:

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	year
count	342.000000	342.000000	342.000000	342.000000	344.000000
mean	43.921930	17.151170	200.915205	4201.754386	2008.029070
std	5.459584	1.974793	14.061714	801.954536	0.818356
min	32.100000	13.100000	172.000000	2700.000000	2007.000000
25%	39.225000	15.600000	190.000000	3550.000000	2007.000000
50%	44.450000	17.300000	197.000000	4050.000000	2008.000000
75%	48.500000	18.700000	213.000000	4750.000000	2009.000000
max	59.600000	21.500000	231.000000	6300.000000	2009.000000

La función anterior `describe()` nos ayuda a obtener algunos parámetros de nuestro conjunto de datos. **Hay que fijarnos que solo incluye los datos numéricos**, así que procederemos a incluir de todo tipo mediante el argumento `include='all'`.

Así me incluye las variables categóricas. Es decir incluye las que no son continuas o discretas.

In [ ]:
```
preprocessed_penguins_df.describe(include='all')
```

Out[ ]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
count	344	344	342.000000	342.000000	342.000000	342.000000	333	344.000000
unique	3	3	NaN	NaN	NaN	NaN	2	NaN
top	Adelie	Biscoe	NaN	NaN	NaN	NaN	male	NaN
freq	152	168	NaN	NaN	NaN	NaN	168	NaN
mean	NaN	NaN	43.921930	17.151170	200.915205	4201.754386	NaN	2008.029070
std	NaN	NaN	5.459584	1.974793	14.061714	801.954536	NaN	0.818356
min	NaN	NaN	32.100000	13.100000	172.000000	2700.000000	NaN	2007.000000
25%	NaN	NaN	39.225000	15.600000	190.000000	3550.000000	NaN	2007.000000
50%	NaN	NaN	44.450000	17.300000	197.000000	4050.000000	NaN	2008.000000
75%	NaN	NaN	48.500000	18.700000	213.000000	4750.000000	NaN	2009.000000
max	NaN	NaN	59.600000	21.500000	231.000000	6300.000000	NaN	2009.000000

La tabla anterior me incluye tanto valores `nulos`, debido a la inclusión de las variables categóricas con variables numéricas.

Para mantener orden lo que hacemos es analizar las variables categóricas y numéricas por separado.

Analizando solo variables numéricas

In [ ]:
```
#Apoyándonos de ser explícitos en la inclusion
#de variables
preprocessed_penguins_df.describe(include=[np.number])
```

Out[ ]:

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	year
count	342.000000	342.000000	342.000000	342.000000	344.000000
mean	43.921930	17.151170	200.915205	4201.754386	2008.029070
std	5.459584	1.974793	14.061714	801.954536	0.818356
min	32.100000	13.100000	172.000000	2700.000000	2007.000000
25%	39.225000	15.600000	190.000000	3550.000000	2007.000000
50%	44.450000	17.300000	197.000000	4050.000000	2008.000000
75%	48.500000	18.700000	213.000000	4750.000000	2009.000000
max	59.600000	21.500000	231.000000	6300.000000	2009.000000

Analizando solo variables categóricas

In [ ]:
```
# Apoyándonos de manera explicita en la inclusion
# de variables
preprocessed_penguins_df.describe(include=object)
```

Out[ ]:

	species	island	sex
count	344	344	333
unique	3	3	2
top	Adelie	Biscoe	male
freq	152	168	168

Pero podemos obtenerlo de otra forma un poco más especial.

In [ ]:
```
(
    preprocessed_penguins_df
    .astype(
        {
            # Convierte la columna 'species' a tipo categórico
            'species': 'category',
            # Convierte la columna 'island' a tipo categórico
            'island': 'category',
            # Convierte la columna 'sex' a tipo categórico
            'sex': 'category'
        }
    )
    # Proporciona un resumen estadístico de las columnas categóricas
    .describe(include=['category', object])
)
```

Out[ ]:

	species	island	sex
count	344	344	333
unique	3	3	2
top	Adelie	Biscoe	male
freq	152	168	168

El código proporcionado realiza los siguientes pasos en un DataFrame llamado `preprocessed_penguins_df`:

1. **Convertir Columnas a Categorías:**

python Copy code

```
preprocessed_penguins_df.astype(
    {
        'species': 'category',
        'island': 'category',
        'sex': 'category'
    }
)
```

Este paso convierte las columnas `species`, `island` y `sex` del DataFrame a tipo `category`. En pandas, los tipos de datos categóricos se usan para datos que tienen un número limitado y generalmente fijo de posibles valores. Esto puede ayudar a reducir el uso de memoria y acelerar ciertas operaciones.

2. **Describir Datos Categóricos:**

python Copy code

```
.describe(include=['category', object])
```

La función `describe()` se utiliza para generar estadísticas descriptivas que resumen la tendencia central, la dispersión y la forma de la distribución de un conjunto de datos. Aquí, el método `describe()` se aplica a todas las columnas de tipo `category` y `object` (que en pandas suele ser el tipo de dato para strings). Esto produce un resumen estadístico para las columnas categóricas, incluyendo:

- `count`: el número de valores no nulos.
- `unique`: el número de categorías únicas.
- `top`: la categoría más frecuente.
- `freq`: la frecuencia de la categoría más frecuente.

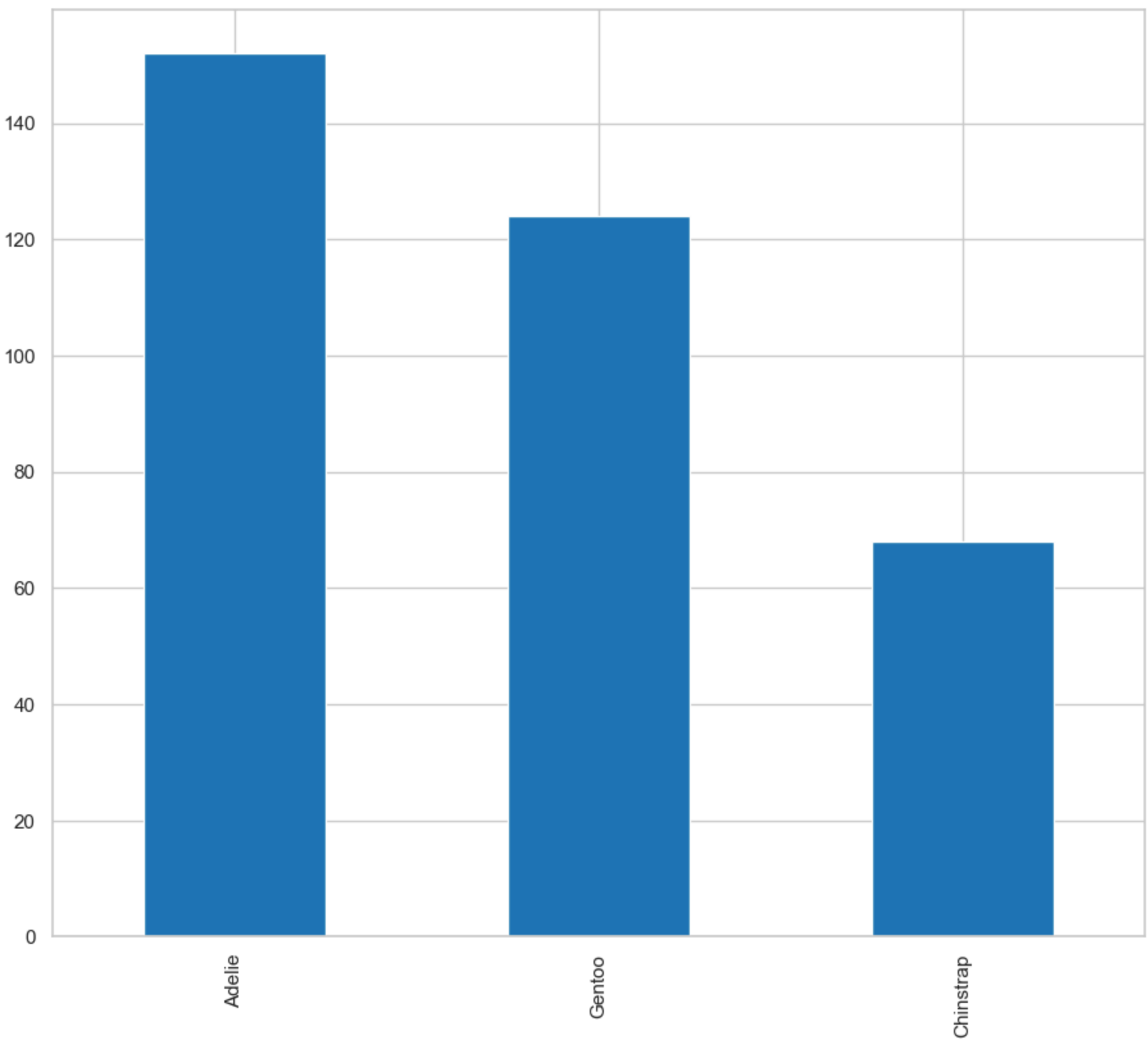
En resumen, el código convierte ciertas columnas a categorías y luego proporciona un resumen estadístico de estas columnas categóricas en el DataFrame. Aquí está el código explicado paso a paso:

¿Cómo puedo visualizar los conteos de categorías?

Pandas

In [ ]:
```
(
    preprocessed_penguins_df
    .species
    .value_counts()
    .plot(kind='bar')

)
```

Out[ ]: <AxesSubplot: >

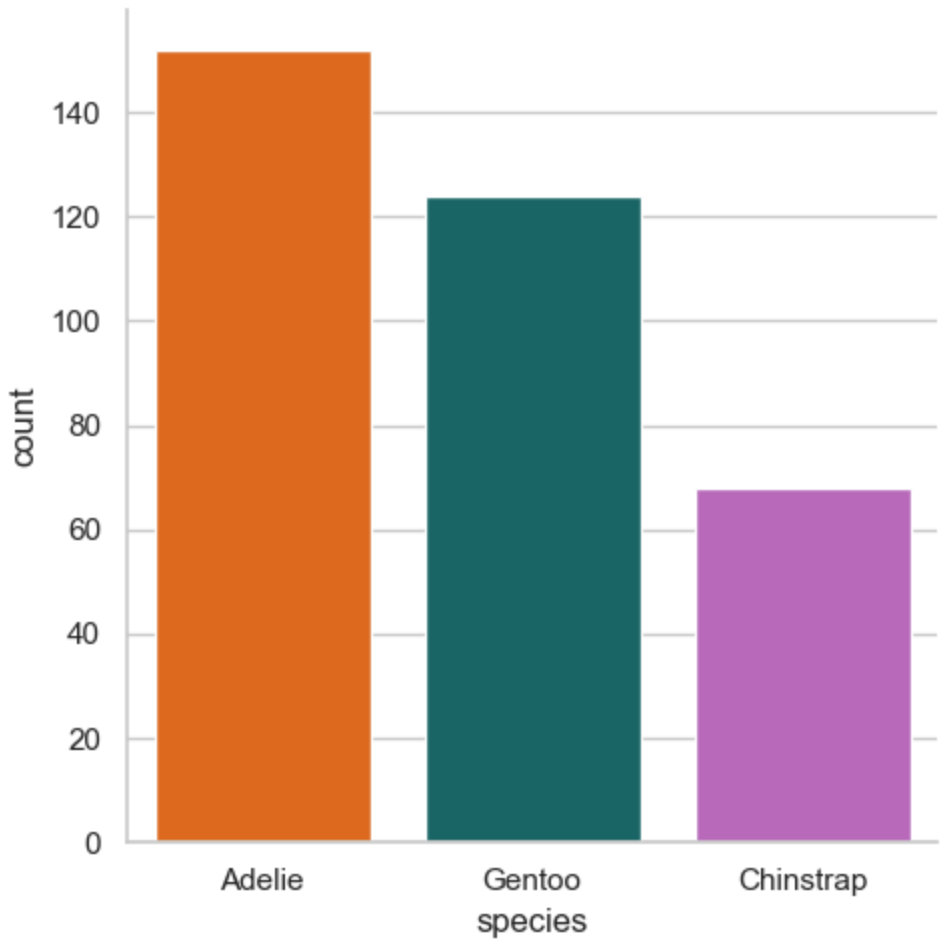


Seaborn

También existen las alternativas de hacerlo con otra biblioteca

In [ ]:
```
sns.catplot(
    data=preprocessed_penguins_df,
    x='species',
    kind='count',
    palette=penguin_color
)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7faa4cf6d360>
```



Otra manera en la cual podríamos graficar es a traves de la función `barplot()`

In [ ]:
```
(
    preprocessed_penguins_df
    .value_counts('species',sort=True)
)
```

Out[ ]:
```
species
Adelie       152
Gentoo       124
Chinstrap     68
dtype: int64
```

El código anterior es para obtener el numero de coincidencias de la especie de cada pingüino, ademas de ordenarlo de mayor a menor. Pero de la anterior forma no lo puede interpretar como una tabla, así que tenemos que resetear el index. Tenemos que poner un `.reset_index()`

In [ ]:
```
(
    preprocessed_penguins_df
    .value_counts('species',sort=True)
    .reset_index(name='counts')
)
```

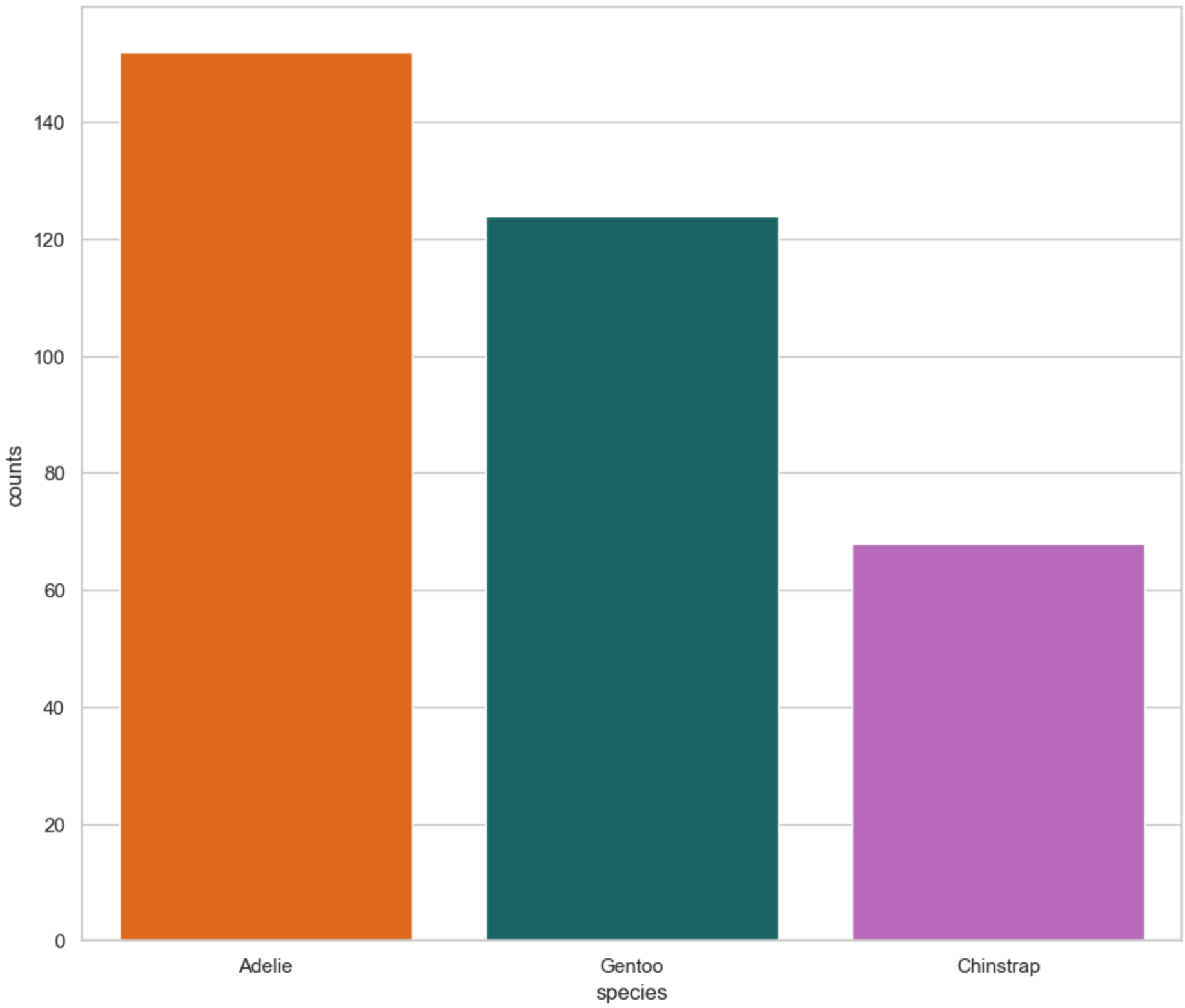
Out[ ]:

	species	counts
0	Adelie	152
1	Gentoo	124
2	Chinstrap	68

De esta forma ya tenemos la representación en tabla, ahora lo único que falta hacer es graficarlo. Mediante una función `lambda` o anónima

In [ ]:
```
(
    preprocessed_penguins_df
    .value_counts('species',sort=True)
    .reset_index(name='counts')
    .pipe(
        lambda df: (
            sns.barplot(
                data=df,
                x='species',
                y='counts',
                palette=penguin_color
            )

                    )
        )
)
```

Out[ ]:
```
<AxesSubplot: xlabel='species', ylabel='counts'>
```



Proporciones

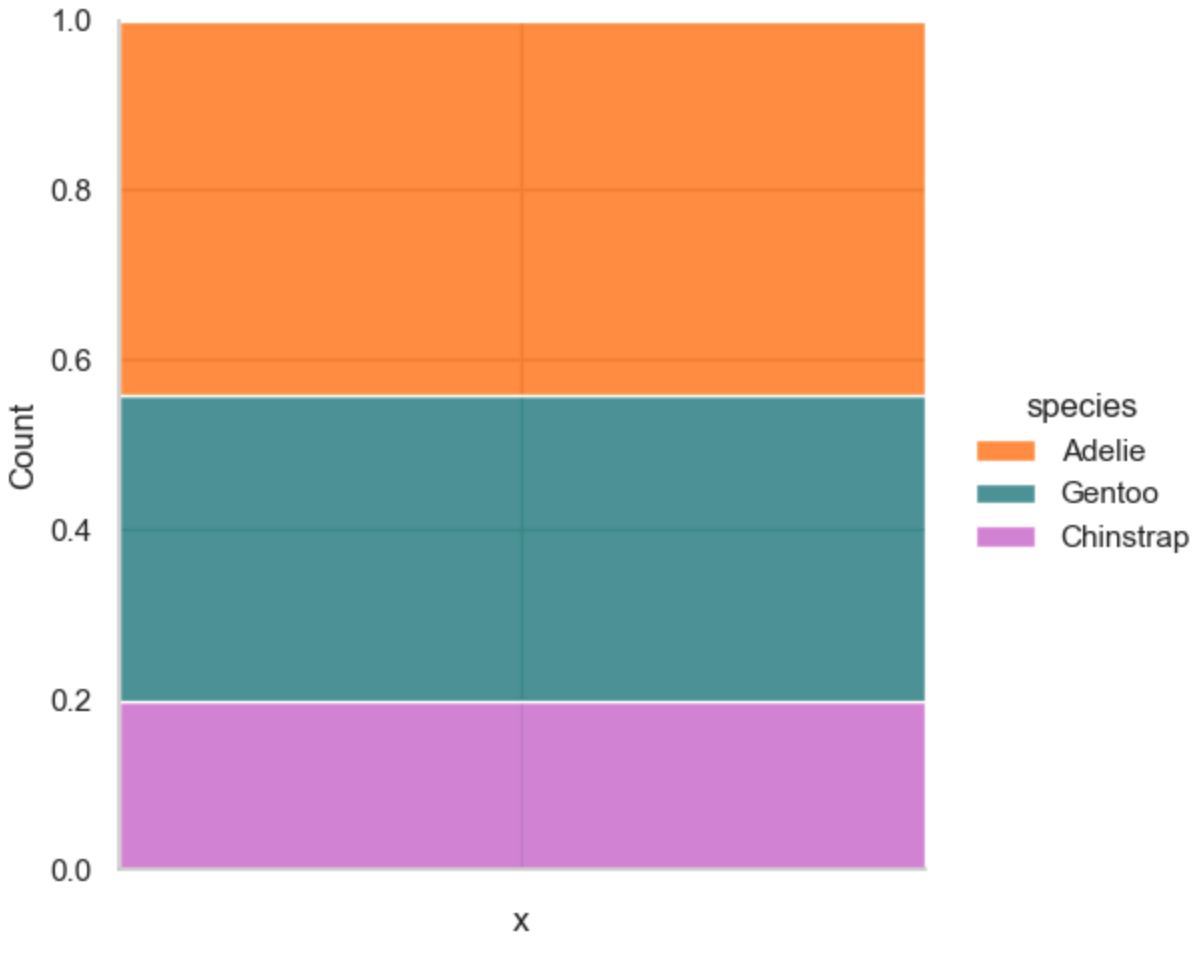
Nosotros en las gráficas pasadas tenemos la frecuencia de aparición, pero ¿cómo podemos ver la proporción de cada valor?

Quizás, queramos saber ¿cuál es la proporción de cada specie respecto a una u otra, o en general?

Para visualizarlo, podemos hacerlo mediante Seaborn.

In [ ]:
```
(
    preprocessed_penguins_df
    .assign(x='') #Añade una columna vacía
    .pipe(
        lambda df: (
            sns.displot(
                data=df,
                x='x',
                hue='species',
                multiple='fill',
                palette=penguin_color
            )

        )
    )
)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7faa45437940>
```



Así ya tengo la proporción de las especies de los pingüinos.

Pero recordemos que tenemos 2 variables categóricas más que son la `Isla` y `Sexo` del pingüino.

Como una extensión puedes explorar las variables de manera individual.

Extendiendo la idea de conteo

La idea de la extensión se le conoce como tabulación cruzada o tablas de contingencia, que permiten mezclar distintas variables categóricas para identificar proporciones o conteos respectivos.

Por ejemplo, tienes un pingüino en la isla, y podemos querer saber cuantos pingüinos 'Adelie' viven en cierta Isla, y así con las demás especies y con otras islas, para saber cual es su proporción. Por el momento continuaremos con una sola variable, ya que es lo que veremos en clases posteriores.

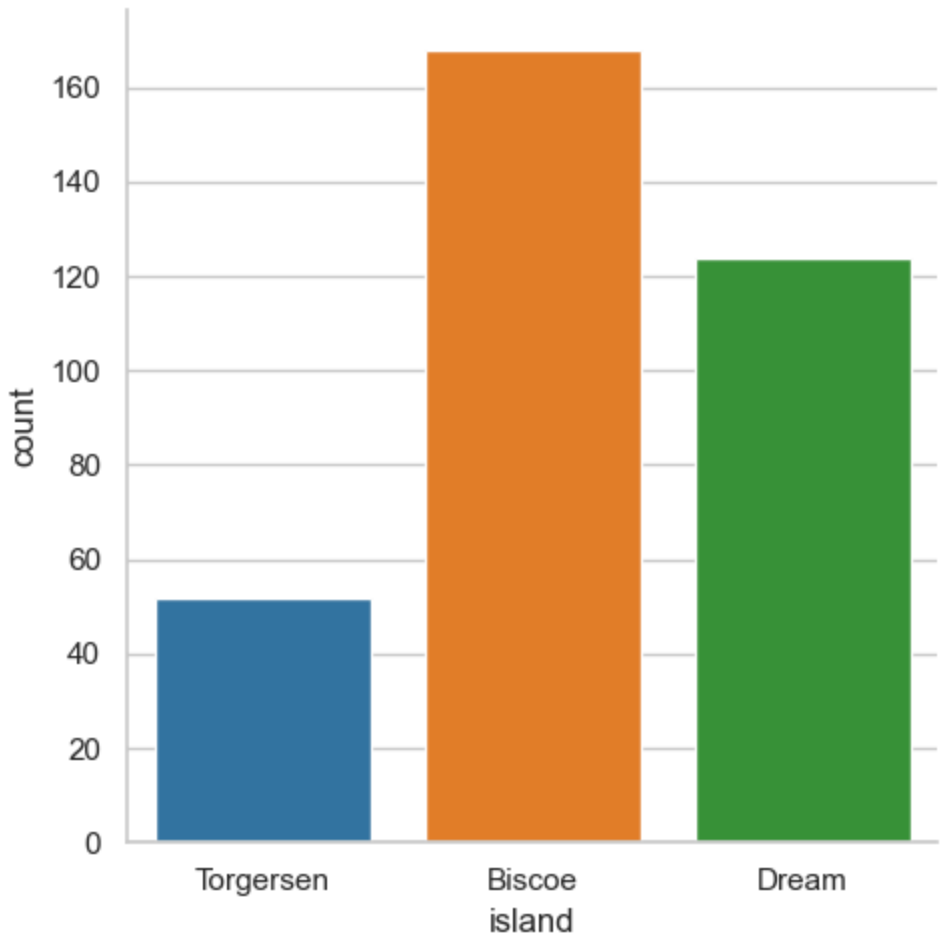
In [ ]:
```
#Veamos que variables
#categóricas tenemos nuevamente
preprocessed_penguins_df.describe(
    include=object
)
```

Out[ ]:

	species	island	sex
count	344	344	333
unique	3	3	2
top	Adelie	Biscoe	male
freq	152	168	168

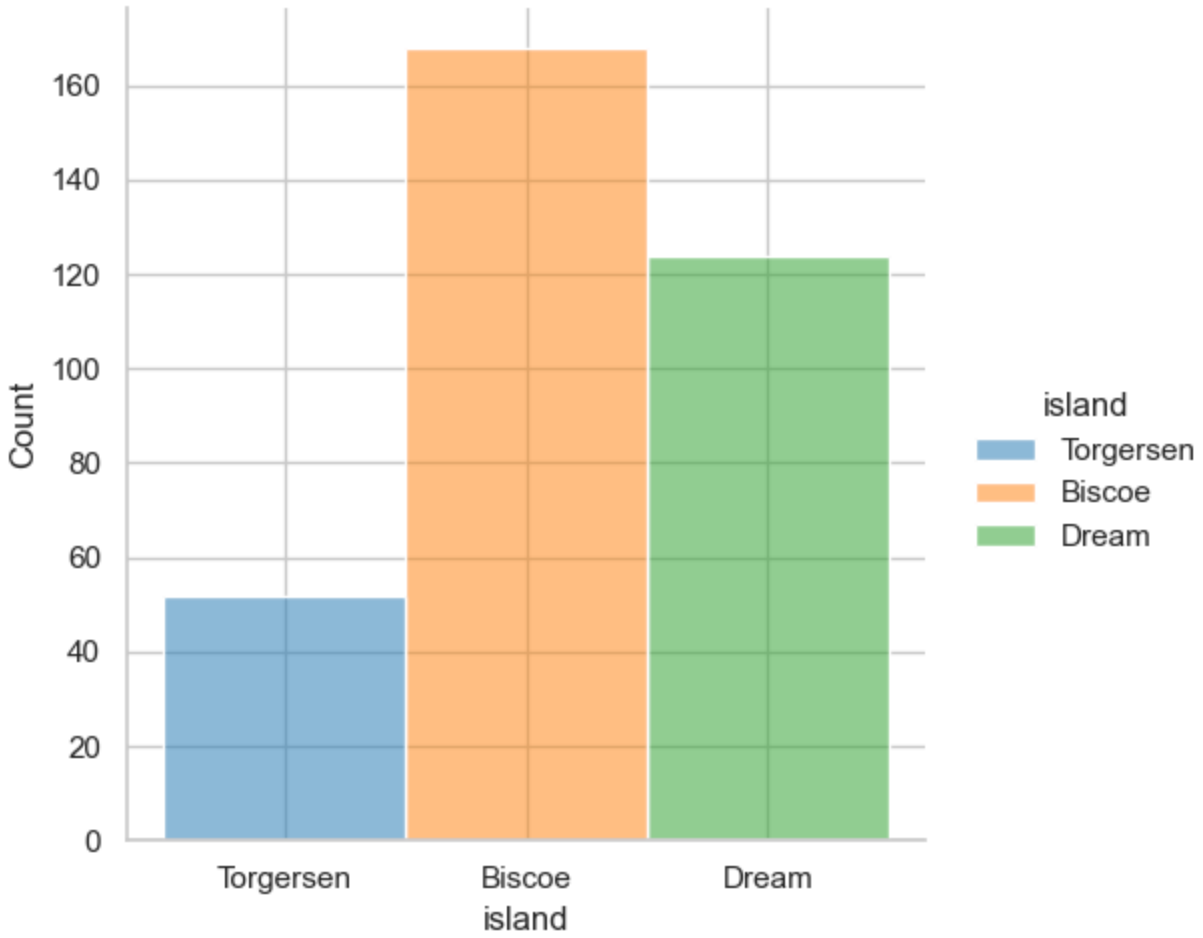
In [ ]:
```
sns.catplot(
    data=preprocessed_penguins_df,
    x='island',
    kind='count',
    hue_order='island',
)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7faa453dbd60>
```



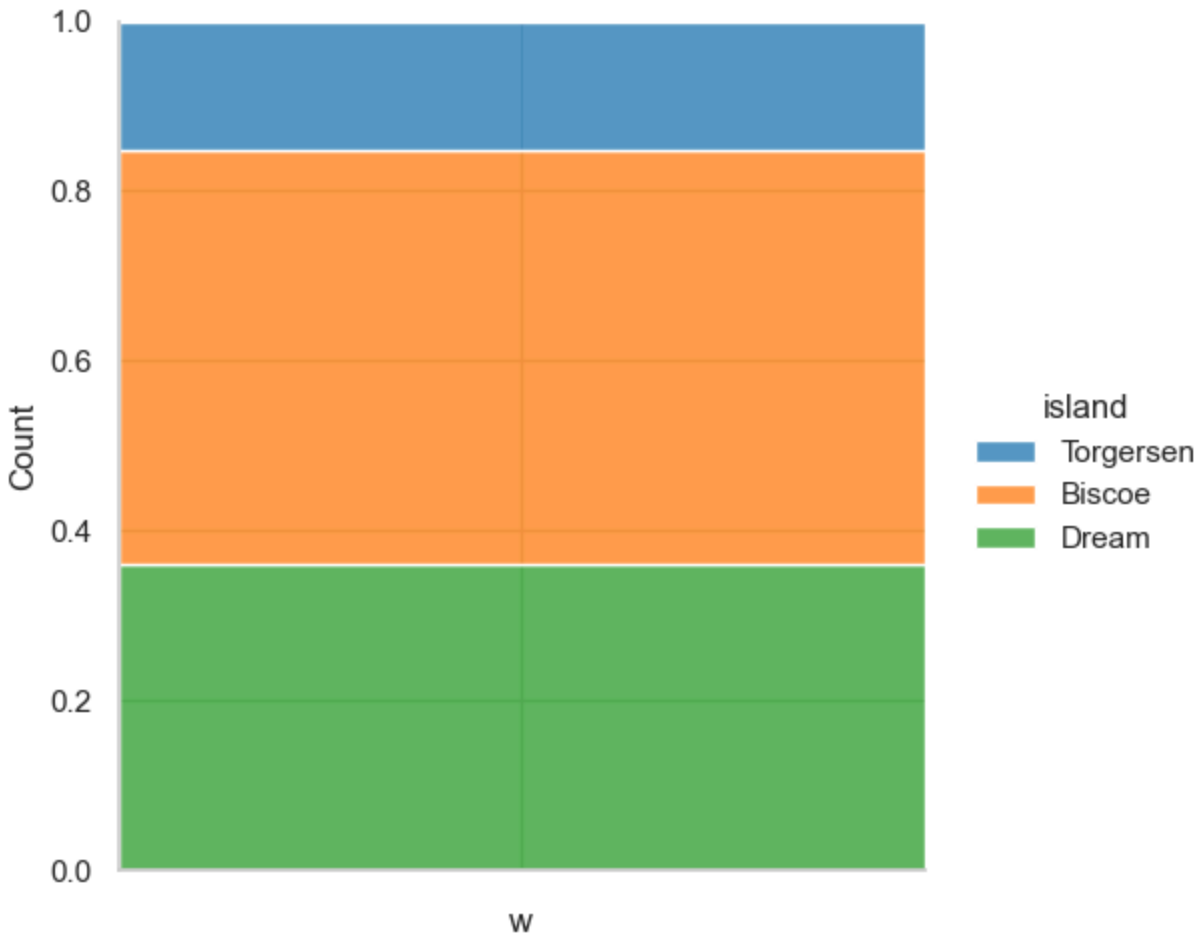
In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='island',
    hue='island',
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa44c3d870>



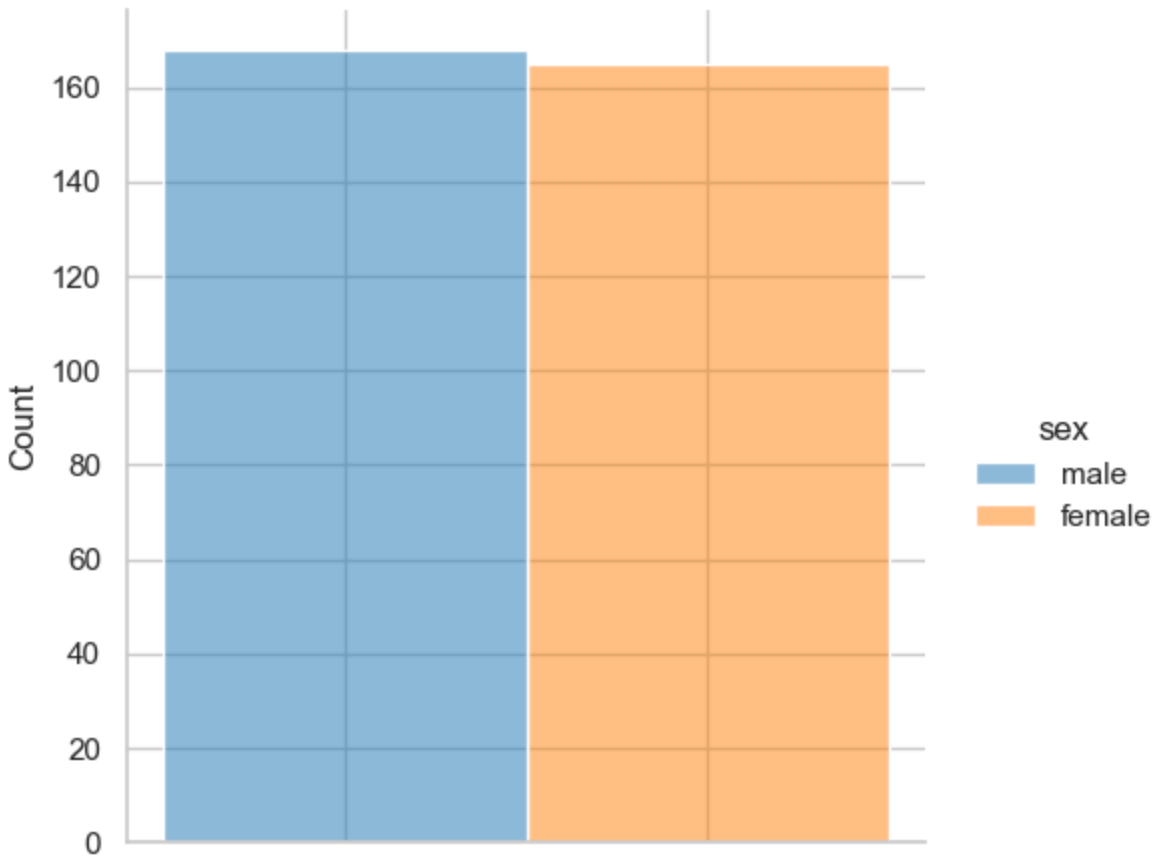
In [ ]:
```
(
    preprocessed_penguins_df
    .assign(w='') #Añade una columna vacía
    .pipe(
        lambda df: (
            sns.displot(
                data=df,
                x='w',
                hue='island',
                multiple='fill',
            )

        )
    )
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa44b5e740>



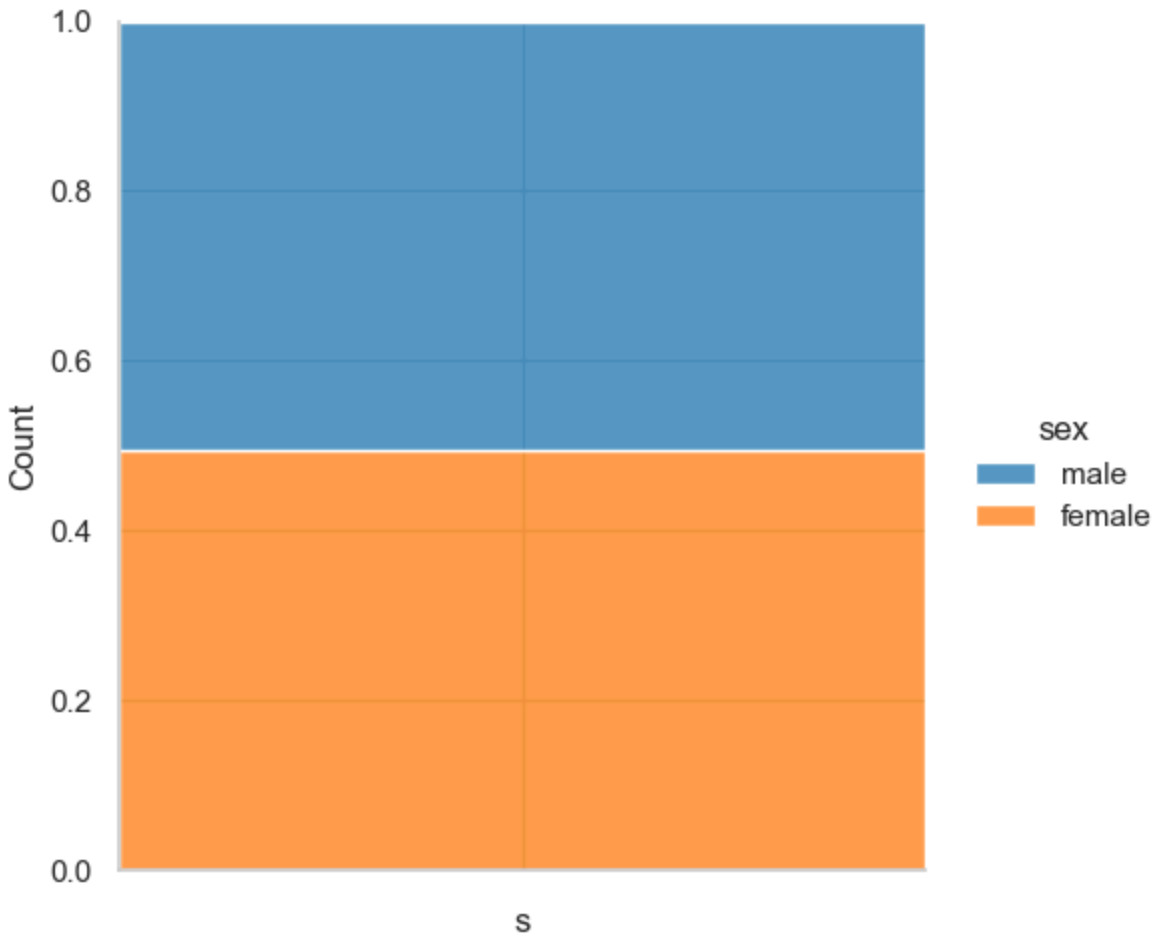
In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='sex',
    hue='sex',
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa3fc3a2c0>



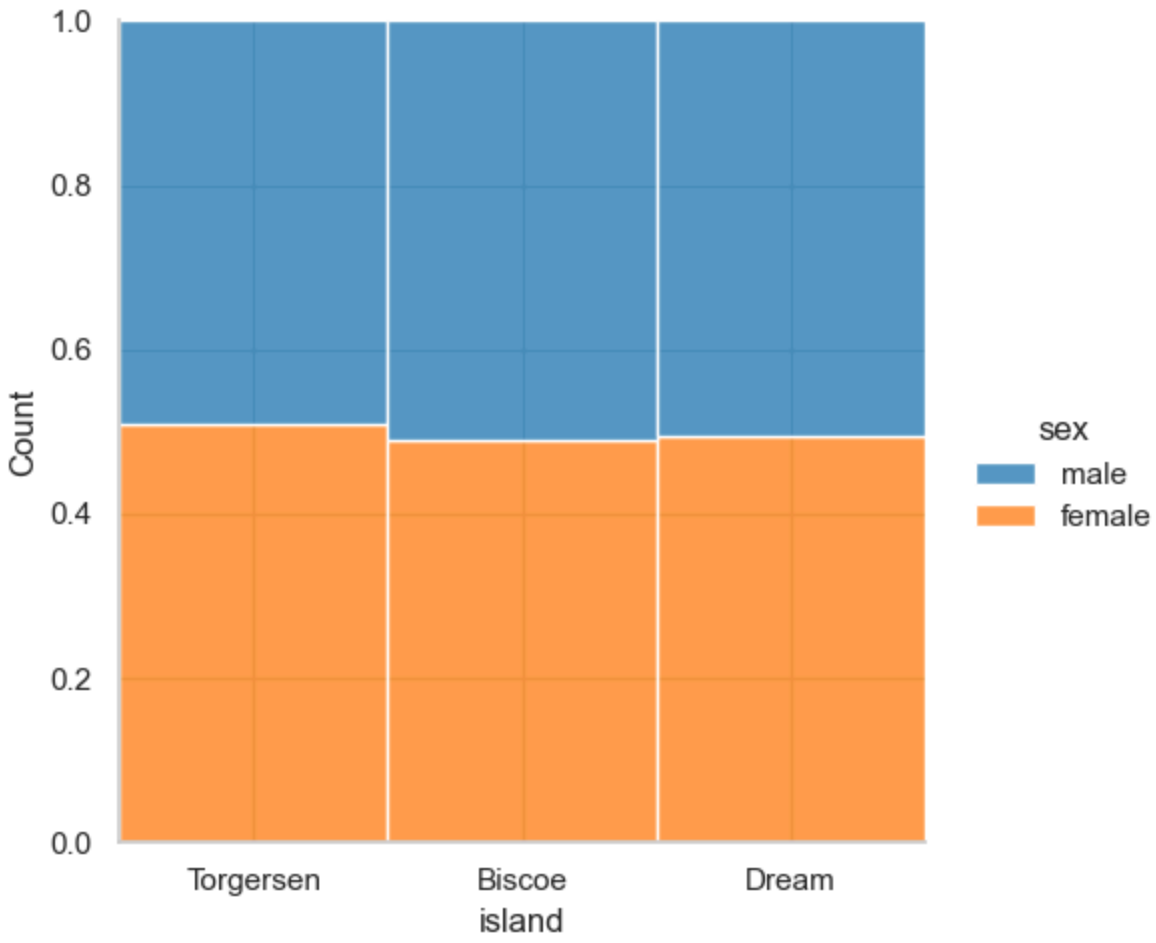
In [ ]:
```
(
    preprocessed_penguins_df
    .assign(s='') #Añade una columna vacía
    .pipe(
        lambda df: (
            sns.displot(
                data=df,
                x='s',
                hue='sex',
                multiple='fill',
            )

        )
    )
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa3fcc2230>



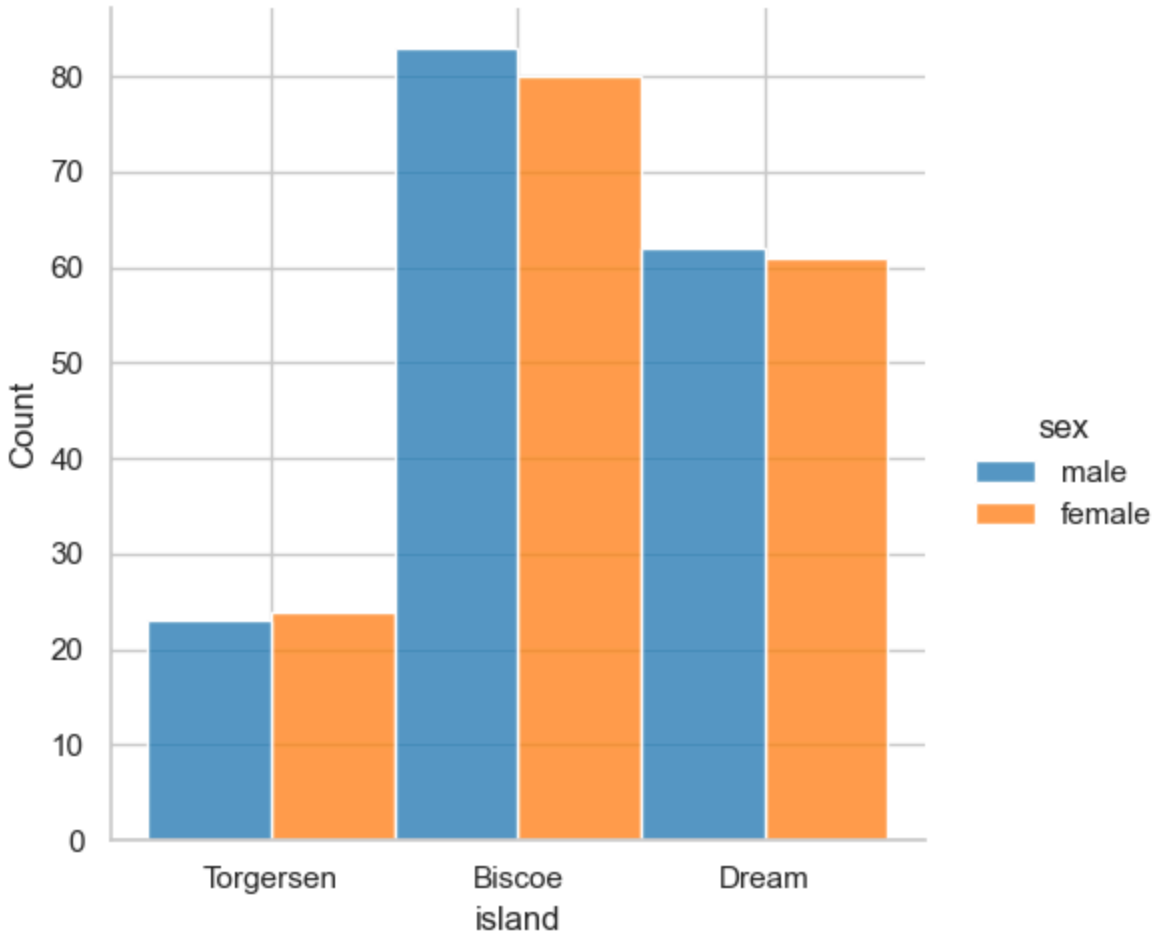
In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='island',
    hue='sex',
    multiple = 'fill'
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa3fcc3d60>



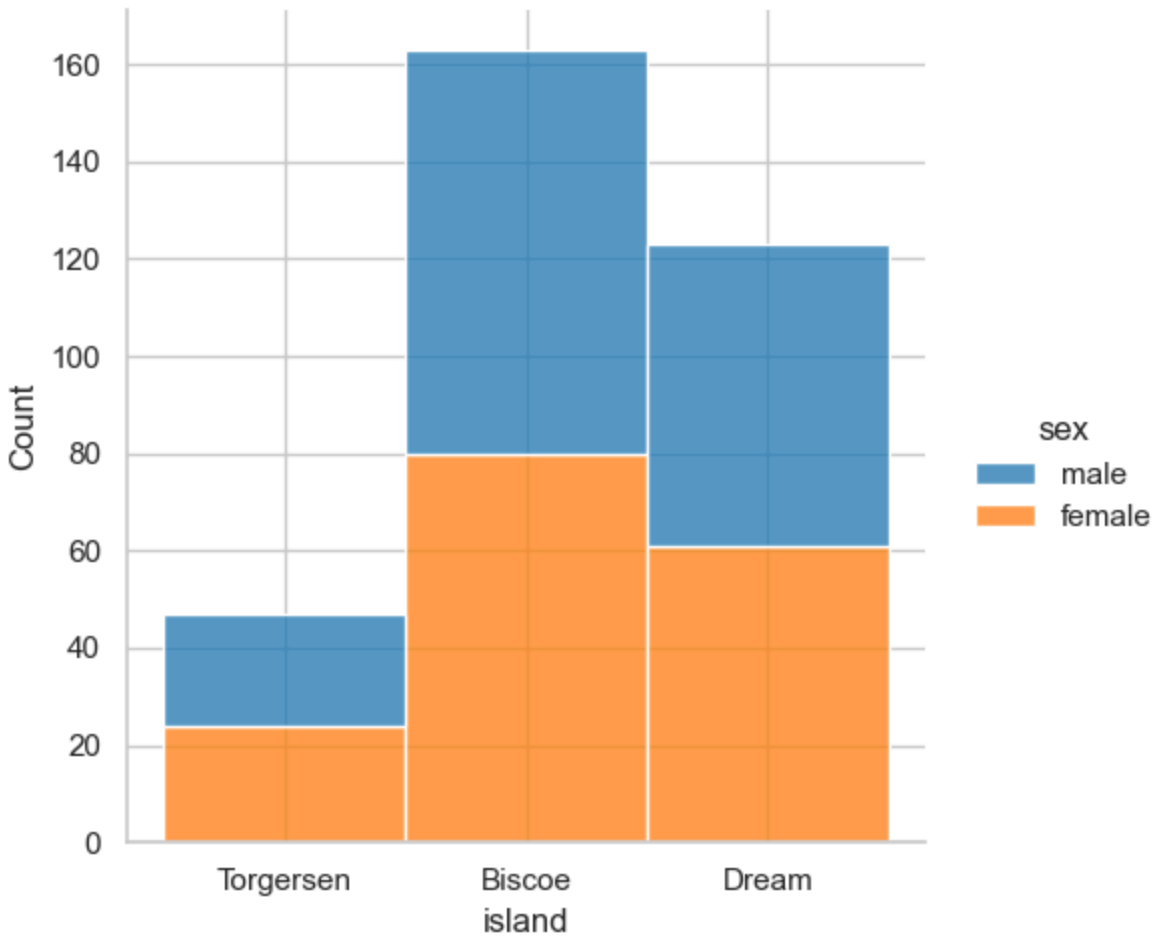
In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='island',
    hue='sex',
    multiple = 'dodge'
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa3fcc3940>



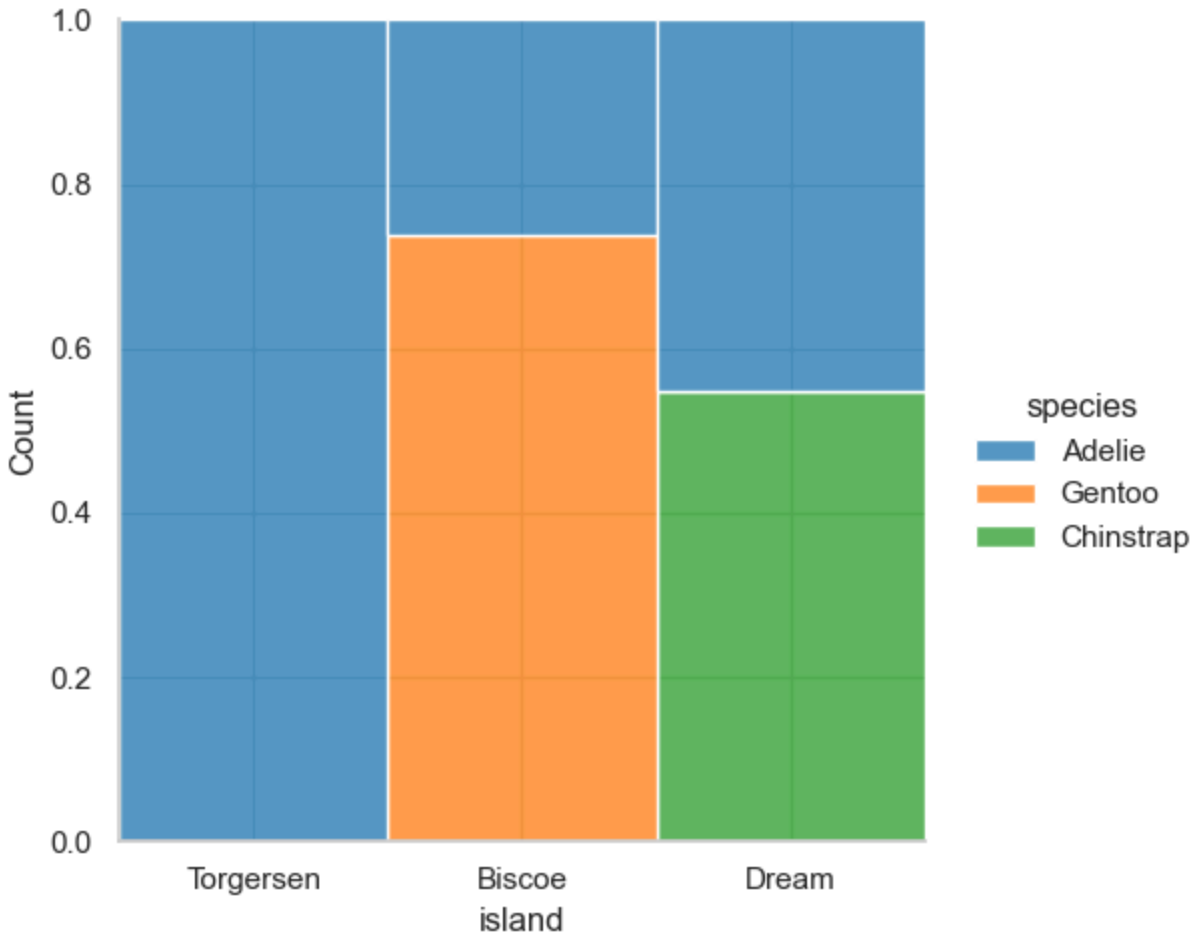
In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='island',
    hue='sex',
    multiple = 'stack'
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa3fc4a620>



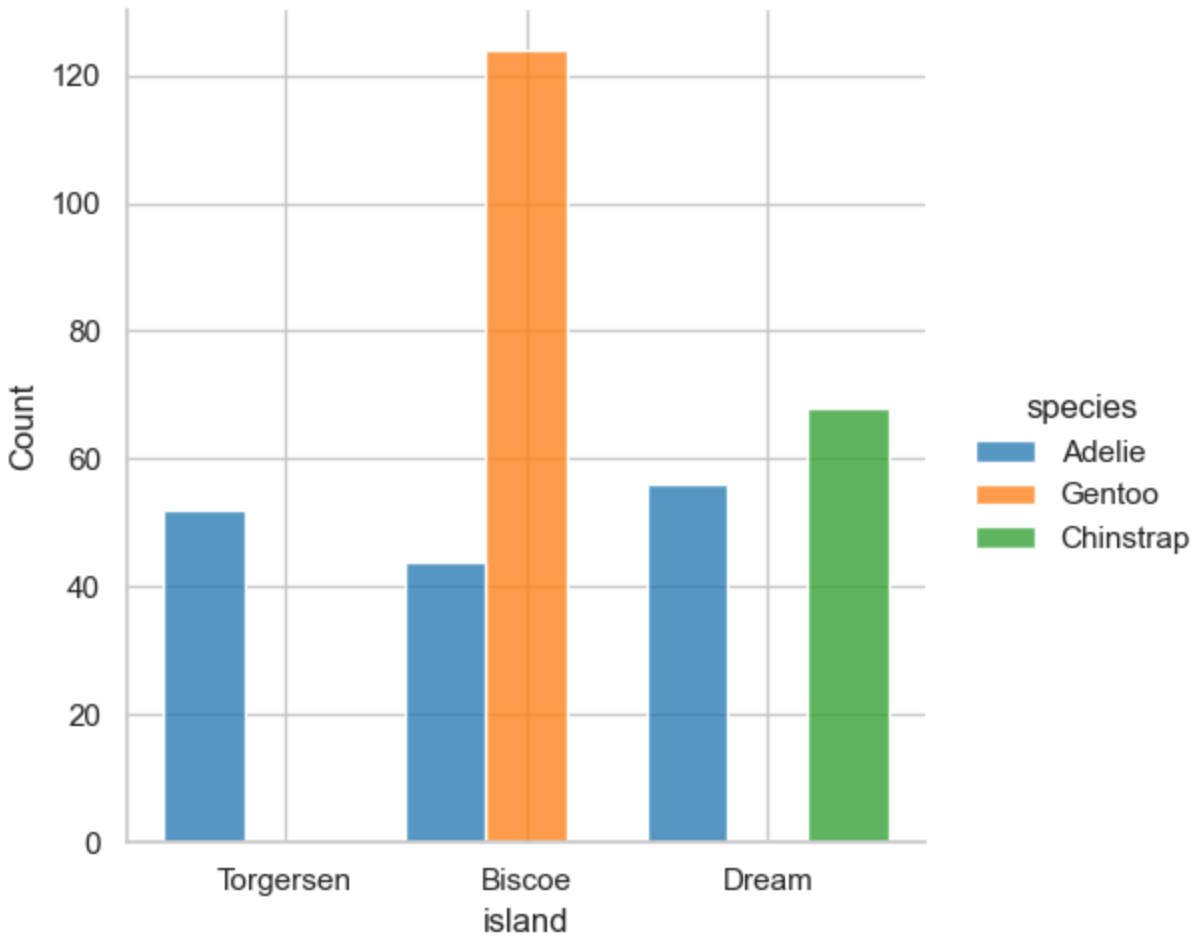
In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='island',
    hue='species',
    multiple = 'fill'
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa44af23b0>



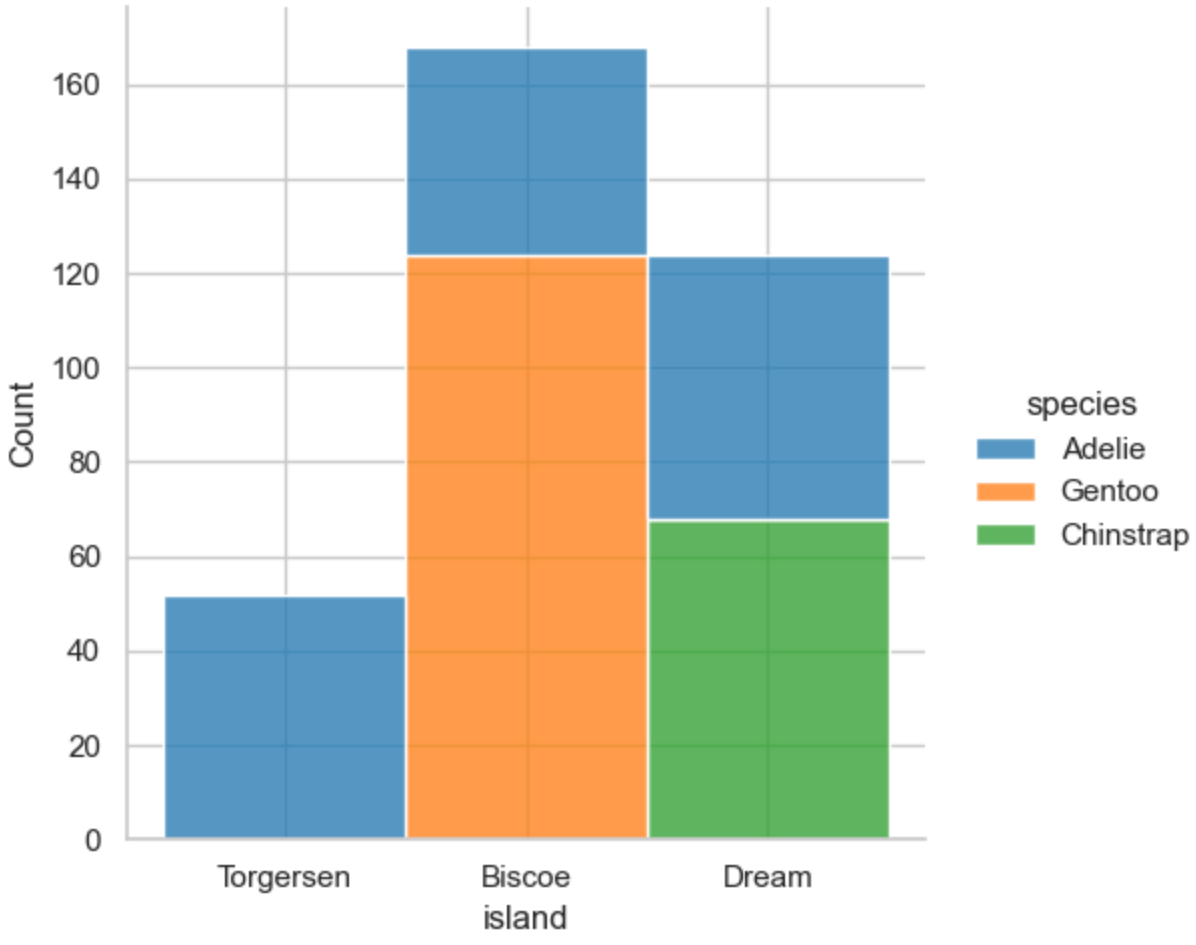
In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='island',
    hue='species',
    multiple = 'dodge'
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa3f9ce680>



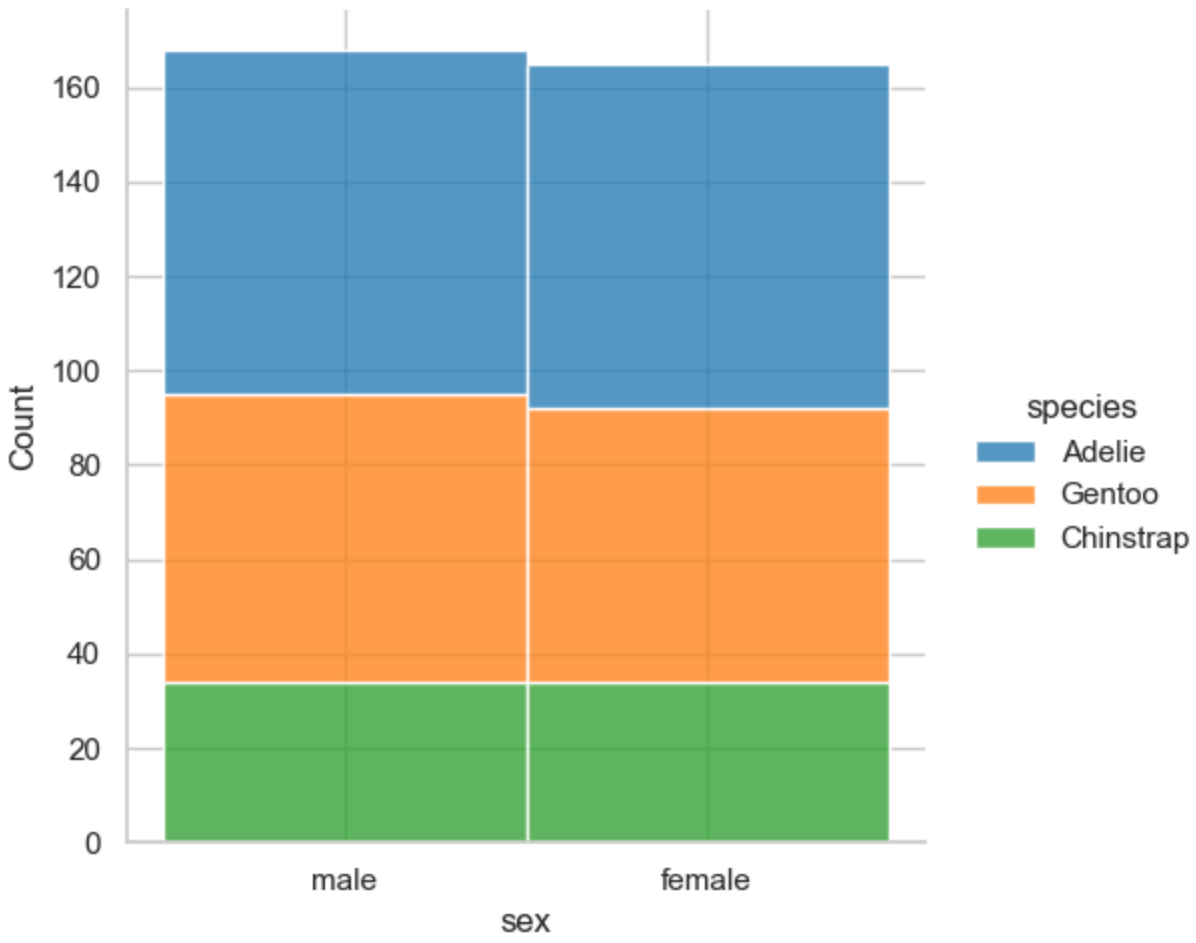
In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='island',
    hue='species',
    multiple = 'stack'
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa3f887be0>



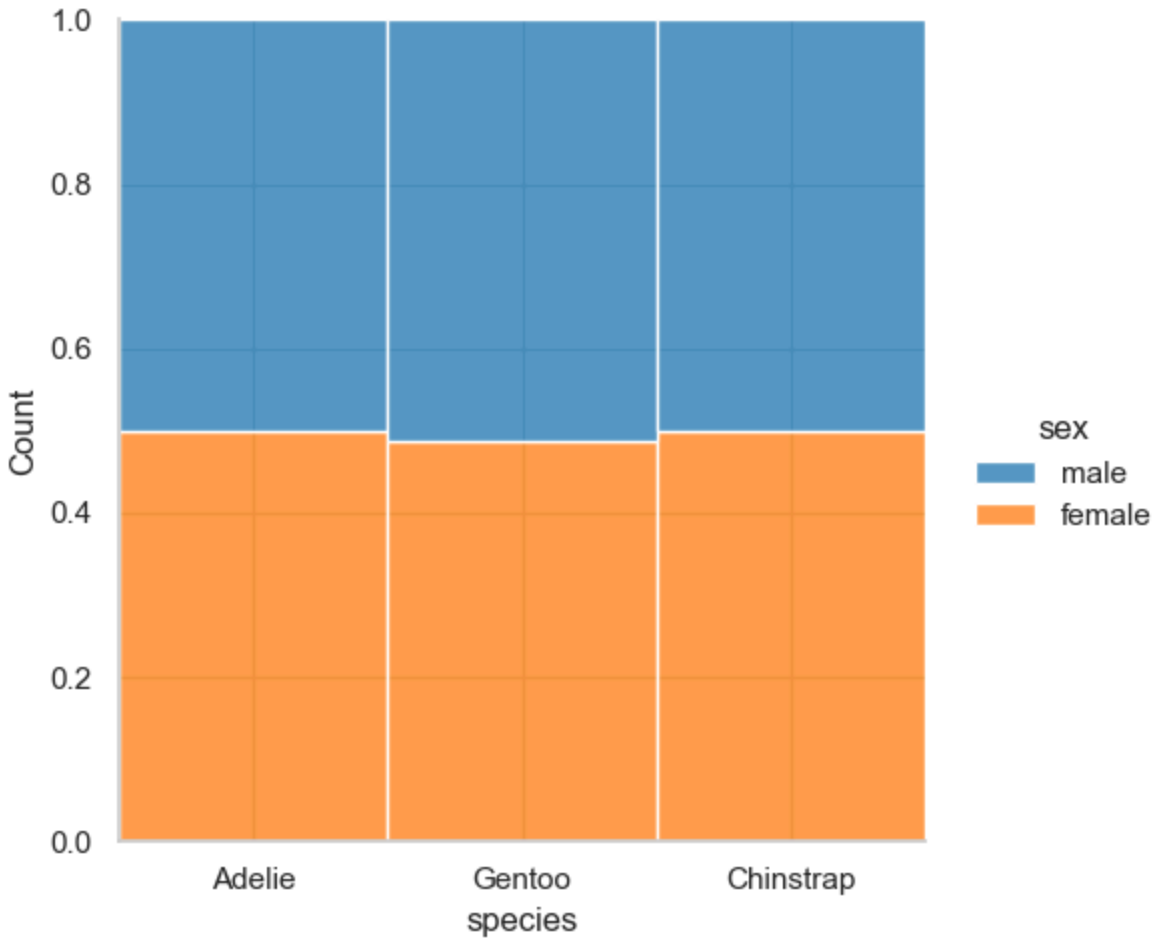
In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='sex',
    hue='species',
    multiple = 'stack'
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa3f72c220>



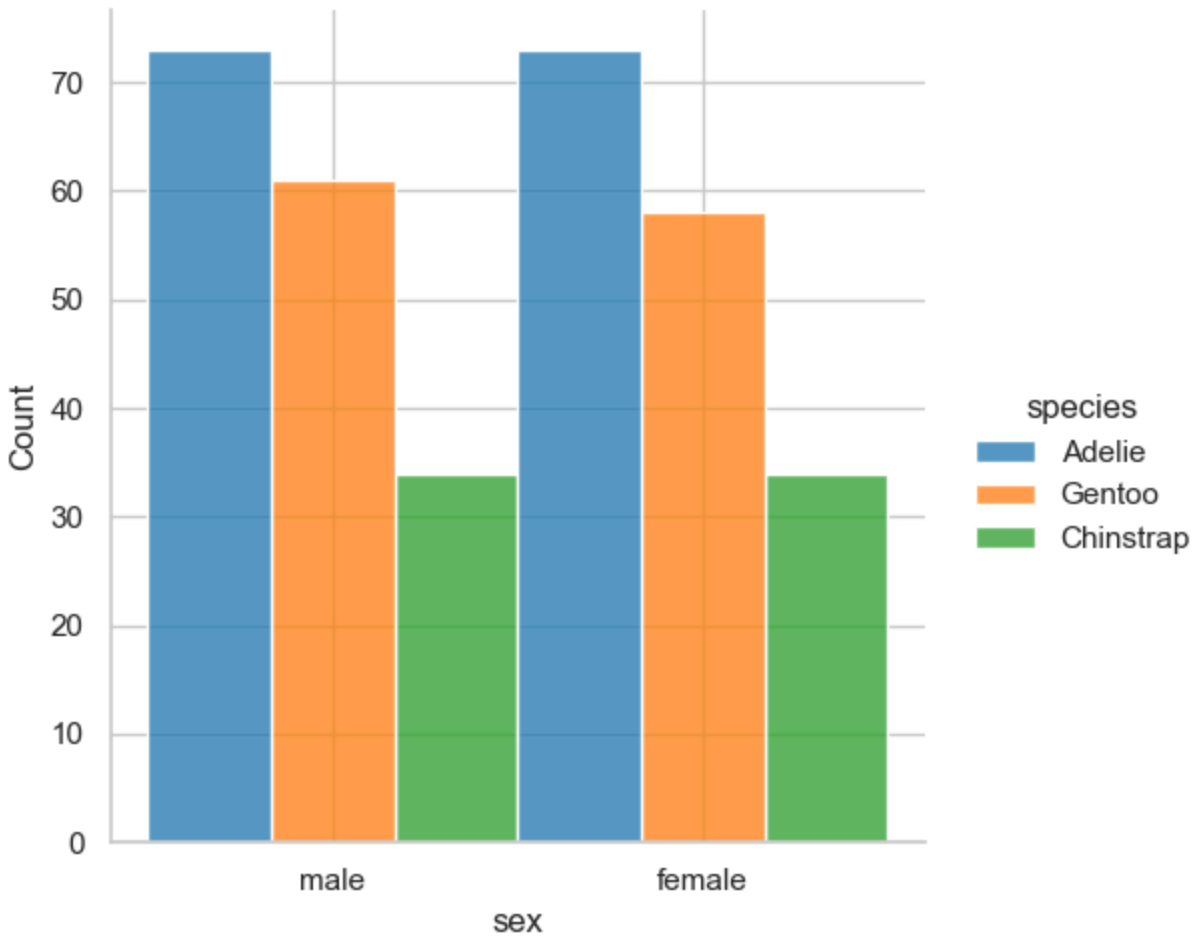
In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='species',
    hue='sex',
    multiple = 'fill'
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa3f491cf0>



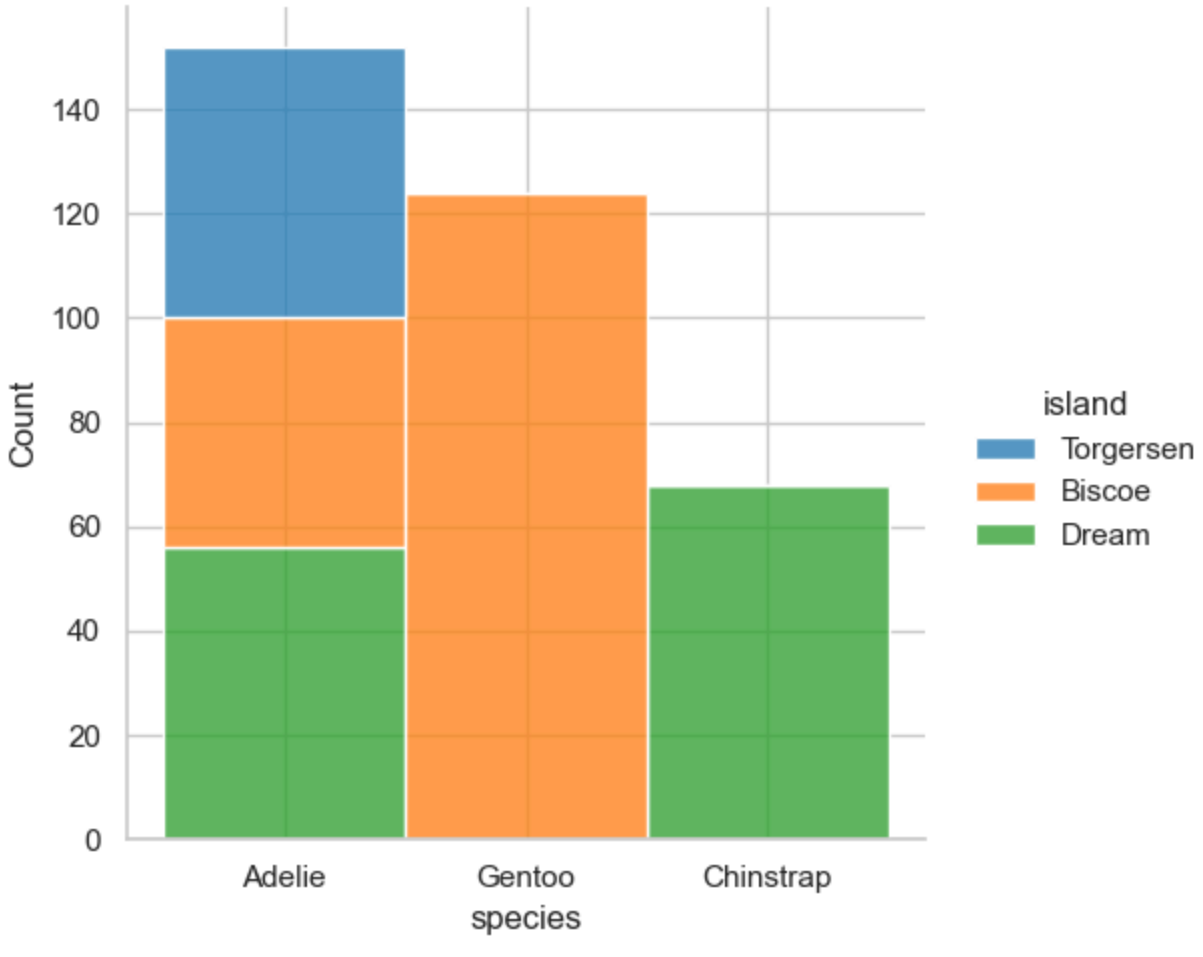
In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='sex',
    hue='species',
    multiple = 'dodge'
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa447e1f60>



In [ ]:
```
sns.displot(
    data=preprocessed_penguins_df,
    x='species',
    hue='island',
    multiple = 'stack'
)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7faa3f64f6d0>


140
120
100
80
60
40
20
0
Count
Adelie
Gentoo
Chinstrap
species
island
Torgersen
Biscoe
Dream