

1. Contexto

En este conjunto de datos relacionados con los pingüinos implica examinar y visualizar los patrones y relaciones presentes en los datos. A continuación, se busca respuesta a las siguientes preguntas:

1. ¿Cual es la vida promedio de un pinguino?
2. ¿Viven as las hembras o los machos?
3. ¿La altura es un rasgo distintivo del sexo?
4. ¿Cual es la proporción altura/ancho de los picos?
5. ¿Qué tipo de datos son las variables del conjunto de datos?
6. ¿Cuántas variables de cada tipo de dato tenemos en el conjunto de datos?
7. ¿Cuántas observaciones y variables tenemos en el conjunto de datos?
8. ¿Existen valores nulos explícitos en el conjunto de datos?
9. ¿De tener observaciones con valores nulos? ¿Cuántas tenemos por cada variable?
10. ¿Cuántos valores nulos tenemos en el total en el conjunto de datos?

2. El dataset después de la limpieza

El dataset ha sido limpiado que dando 333 filas y 9 columnas

3. Lectura del dataset

```
In [ ]: # Importando Librerias
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```


```
In [ ]: # Lectura del dataset
ruta = "../results/dataset_penguins_clean.csv"
data = pd.read_csv(ruta)
```

```
In [ ]: # Mostrando el dataset
print(data.shape)
data.head()
```

(333, 9)

Out[]:

| | rowid | species | island | bill_length_mm | bill_depth_mm | flipper_length_mm | body_mas |
|---|-------|---------|-----------|----------------|---------------|-------------------|----------|
| 0 | 1 | Adelie | Torgersen | 39.1 | 18.7 | 181.0 | 375 |
| 1 | 2 | Adelie | Torgersen | 39.5 | 17.4 | 186.0 | 380 |
| 2 | 3 | Adelie | Torgersen | 40.3 | 18.0 | 195.0 | 325 |
| 3 | 5 | Adelie | Torgersen | 36.7 | 19.3 | 193.0 | 345 |
| 4 | 6 | Adelie | Torgersen | 39.3 | 20.6 | 190.0 | 365 |



4. Análisis exploratorio (EDA)

La idea es usar herramientas estadísticas y de visualización para:

- Crear un mapa mental del set de datos (entenderlo)
- Empezar a encontrar respuestas a las preguntas planteadas inicialmente.

Se llevará a cabo estas fases:

1. Análisis de cada variable de manera individual
2. Análisis univariado: relación de cada variable predictora con la variable a predecir
3. Análisis bivariado: relación de pares de variables predictoras con la variable a predecir

4.1 Análisis de cada variable de manera individual

Esto nos permitira entender las características generales de cada variable del dataset.

```
In [ ]: # Primero que ndad, veremos las variables categoricas.
# Son del tipo: Dtype=object()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 333 entries, 0 to 332
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   rowid                 333 non-null   int64
1   species               333 non-null   object
2   island                333 non-null   object
3   bill_length_mm        333 non-null   float64
4   bill_depth_mm         333 non-null   float64
5   flipper_length_mm     333 non-null   float64
6   body_mass_g           333 non-null   float64
7   sex                   333 non-null   object
8   year                  333 non-null   int64
dtypes: float64(4), int64(2), object(3)
memory usage: 23.5+ KB
```

```
In [ ]: # Veamos las variables categoricas en graficos de barras.
columnas_categoricas = ['species', 'island', 'sex']

# Graficas de barras de conteo
fig, ax = plt.subplots(nrows=len(columnas_categoricas),
                        ncols=1, figsize=(10, 30))
fig.subplots_adjust(hspace=0.5)

for i, col in enumerate(columnas_categoricas):
    sns.countplot(x=col, data=data, ax=ax[i])
    ax[i].set_title(col)
    ax[i].set_xticklabels(ax[i].get_xticklabels(), rotation=30)
```

C:\Users\josetorres\AppData\Local\Temp\ipykernel_14512\552490993.py:12: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.

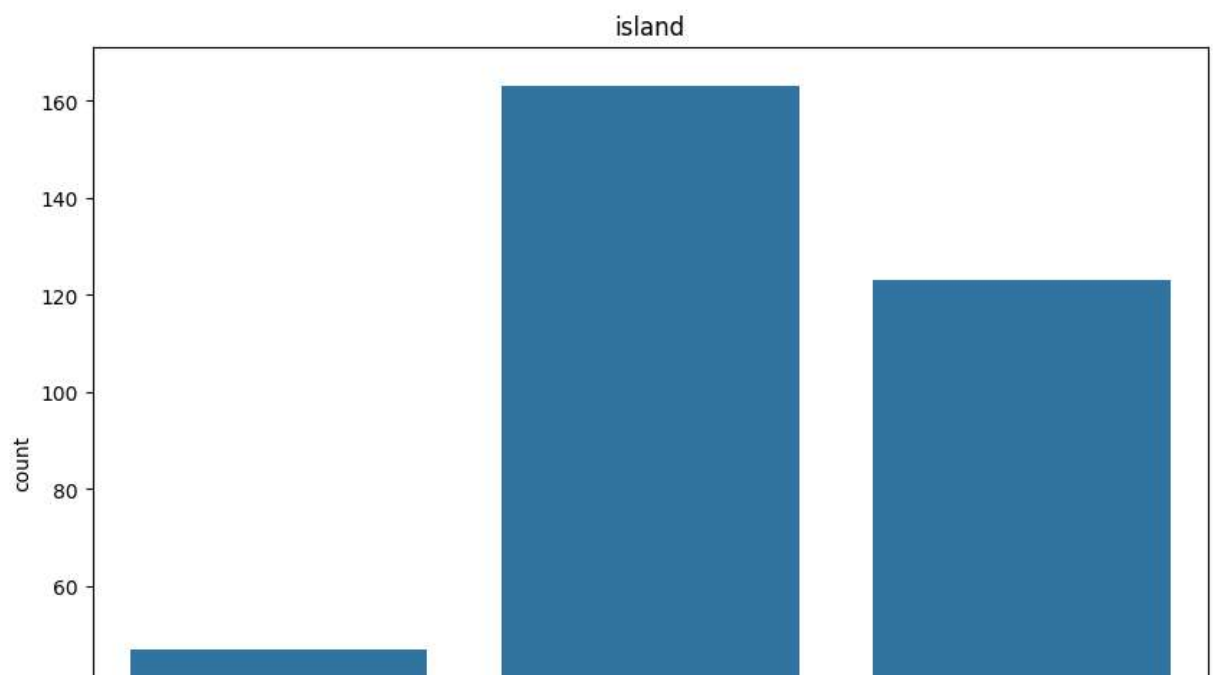
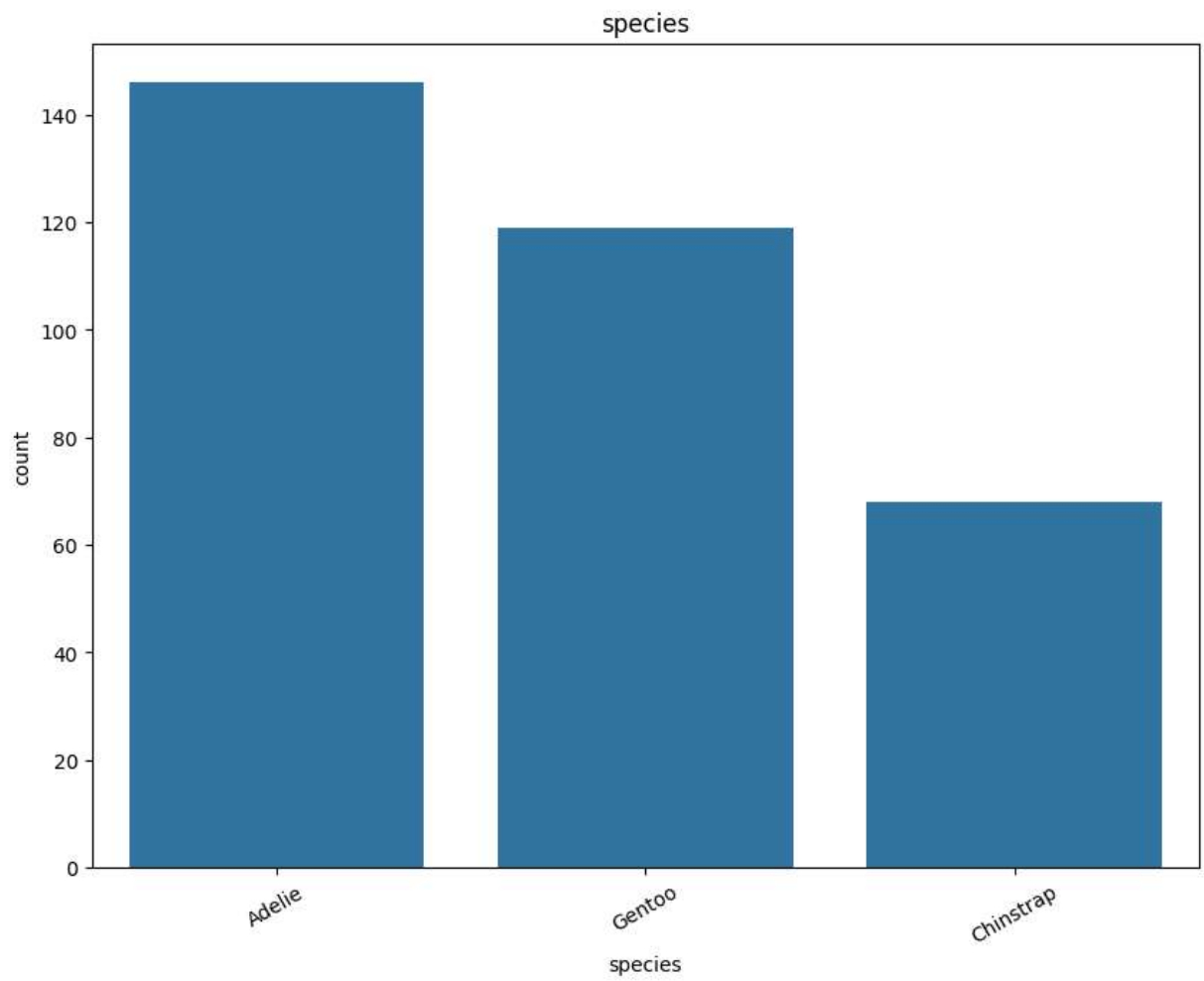
ax[i].set_xticklabels(ax[i].get_xticklabels(), rotation=30)

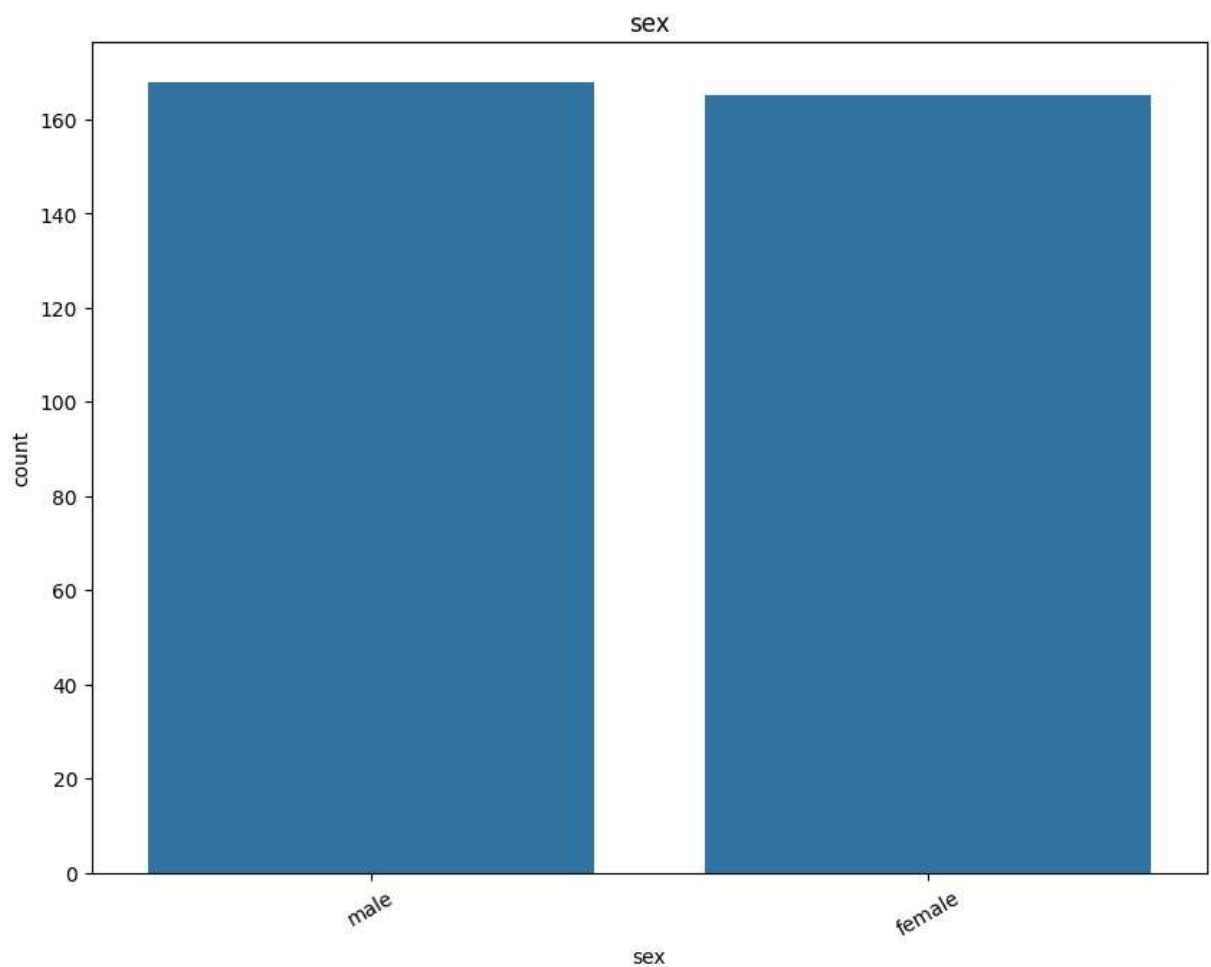
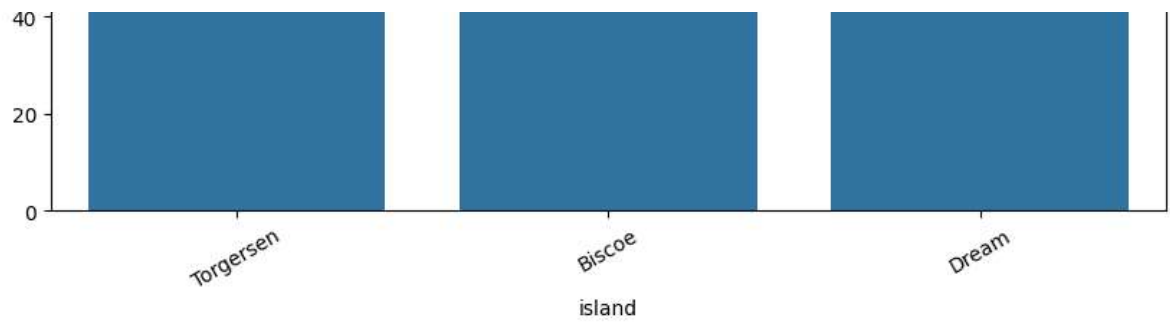
C:\Users\josetorres\AppData\Local\Temp\ipykernel_14512\552490993.py:12: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.

ax[i].set_xticklabels(ax[i].get_xticklabels(), rotation=30)

C:\Users\josetorres\AppData\Local\Temp\ipykernel_14512\552490993.py:12: UserWarning: set_ticklabels() should only be used with a fixed number of ticks, i.e. after set_ticks() or using a FixedLocator.

ax[i].set_xticklabels(ax[i].get_xticklabels(), rotation=30)





```
In [ ]: # Volvamos a ver las variables del dataset
print("Variables del dataset:")
data.info()

# Ahora, extraemos las variables estadísticas descriptivas básicas.
print("\nVariables estadísticas descriptivas básicas:")
data.describe()
```

Variables del dataset:

```
<class 'pandas.core.frame.DataFrame'>
```

RangeIndex: 333 entries, 0 to 332

Data columns (total 9 columns):

| # | Column | Non-Null Count | Dtype |
|---|-------------------|----------------|---------|
| 0 | rowid | 333 non-null | int64 |
| 1 | species | 333 non-null | object |
| 2 | island | 333 non-null | object |
| 3 | bill_length_mm | 333 non-null | float64 |
| 4 | bill_depth_mm | 333 non-null | float64 |
| 5 | flipper_length_mm | 333 non-null | float64 |
| 6 | body_mass_g | 333 non-null | float64 |
| 7 | sex | 333 non-null | object |
| 8 | year | 333 non-null | int64 |

dtypes: float64(4), int64(2), object(3)

memory usage: 23.5+ KB

Variables estadísticas descriptivas básicas:

Out []:

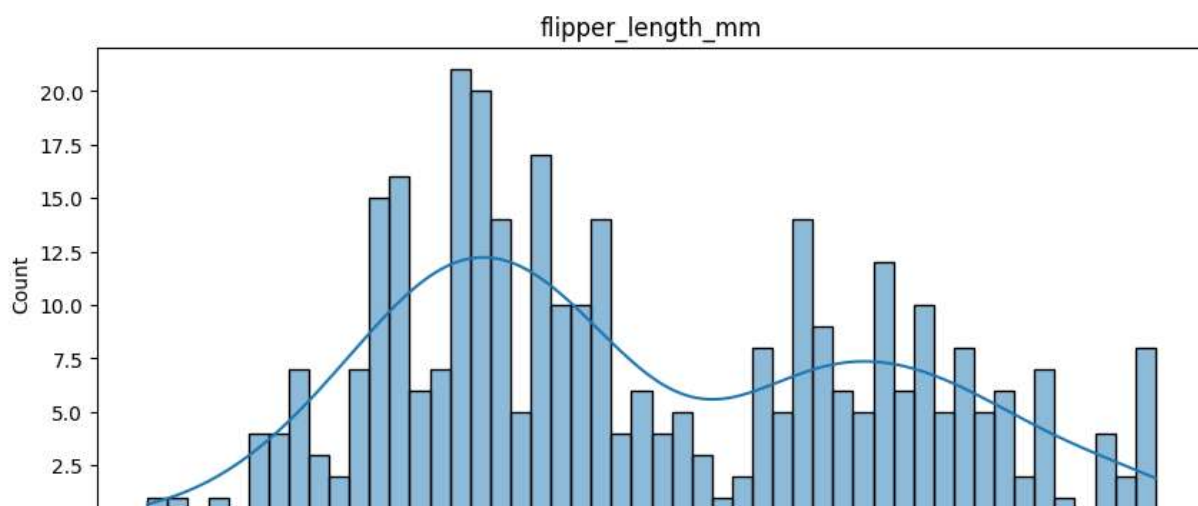
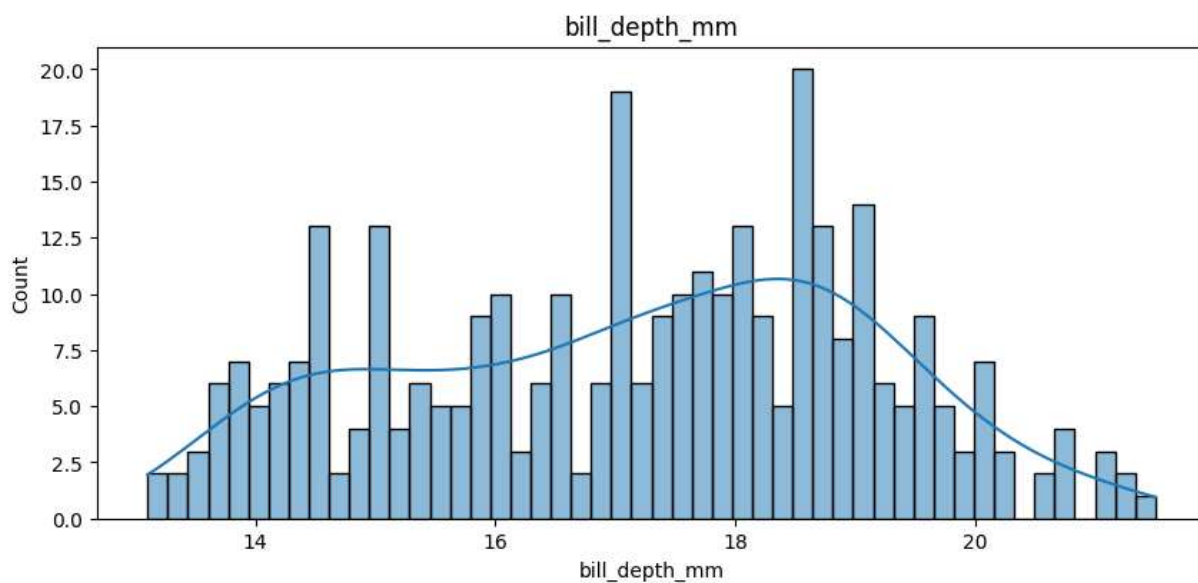
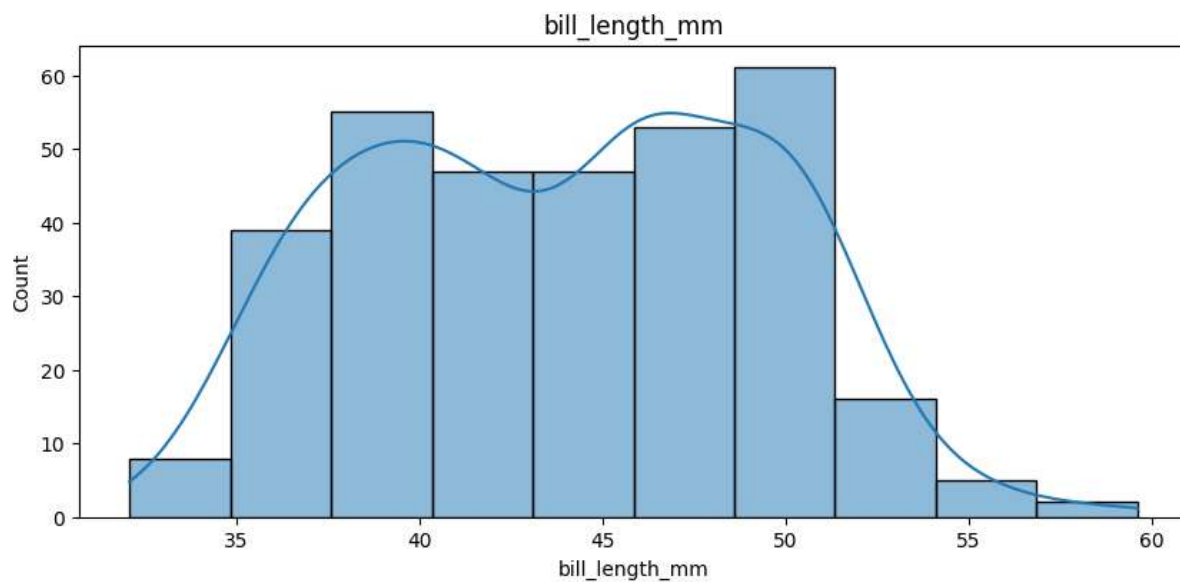
| | rowid | bill_length_mm | bill_depth_mm | flipper_length_mm | body_mass_g | |
|--------------|------------|----------------|---------------|-------------------|-------------|-------------|
| count | 333.000000 | 333.000000 | 333.000000 | 333.000000 | 333.000000 | 333.000000 |
| mean | 174.324324 | 43.992793 | 17.164865 | 200.966967 | 4207.057057 | 2008.000000 |
| std | 98.386547 | 5.468668 | 1.969235 | 14.015765 | 805.215802 | 0.800000 |
| min | 1.000000 | 32.100000 | 13.100000 | 172.000000 | 2700.000000 | 2007.000000 |
| 25% | 90.000000 | 39.500000 | 15.600000 | 190.000000 | 3550.000000 | 2007.000000 |
| 50% | 173.000000 | 44.500000 | 17.300000 | 197.000000 | 4050.000000 | 2008.000000 |
| 75% | 259.000000 | 48.600000 | 18.700000 | 213.000000 | 4775.000000 | 2009.000000 |
| max | 344.000000 | 59.600000 | 21.500000 | 231.000000 | 6300.000000 | 2009.000000 |

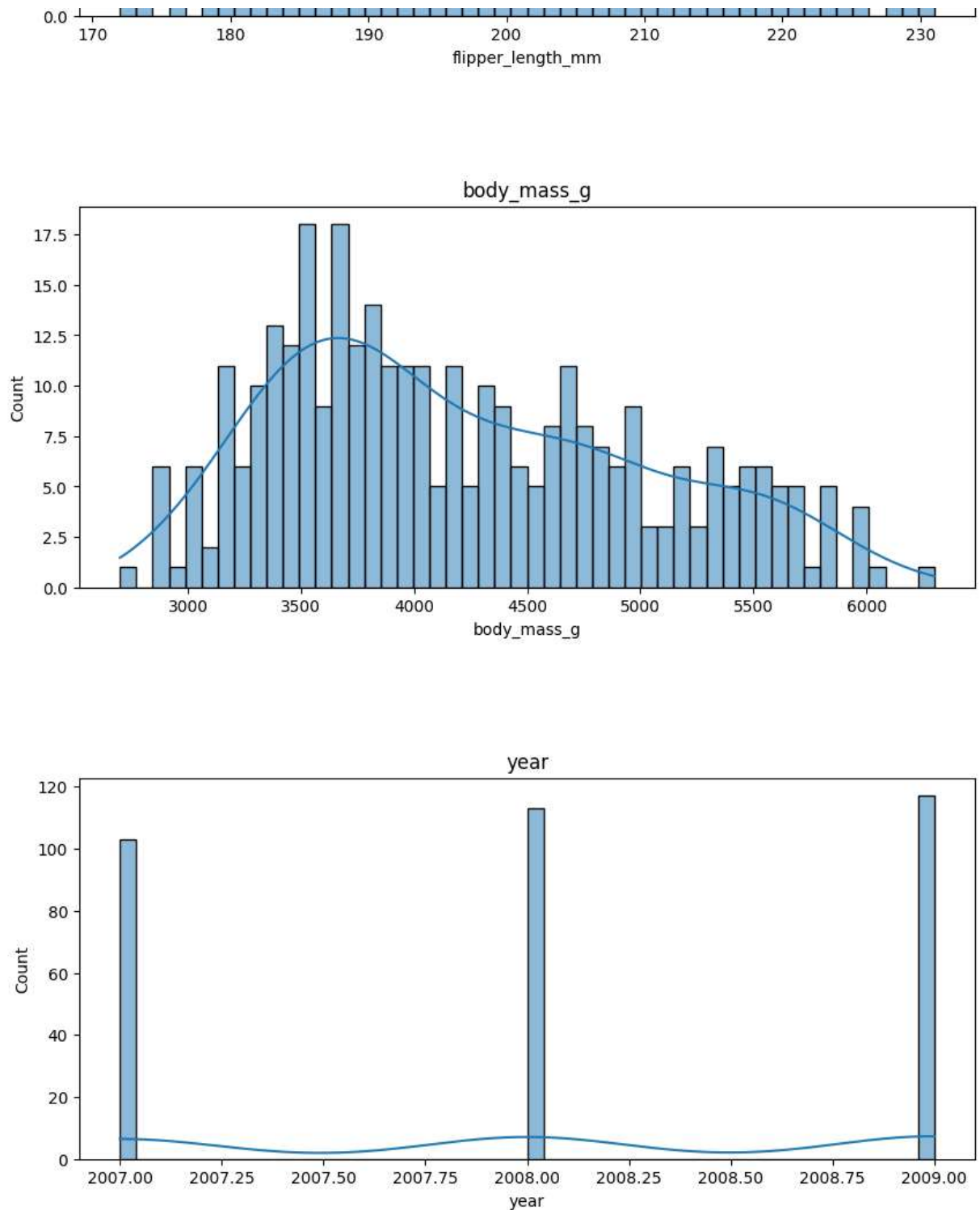
```
In [ ]: # Es mejor graficar estas variables estadísticas.

# Dibujaremos histogramas
columnas_numericas = ['bill_length_mm', 'bill_depth_mm',
                      'flipper_length_mm', 'body_mass_g', 'year']

# Cuantos gráficos se mostrarán
fig, ax = plt.subplots(nrows=5, ncols=1, figsize=(10, 30))
fig.subplots_adjust(hspace=0.5)

for i, col in enumerate(columnas_numericas):
    if col == 'bill_length_mm':
        nbins = 10
    else:
        nbins = 50
    sns.histplot(x=col, data=data, ax=ax[i], bins=nbins, kde=True)
    ax[i].set_title(col)
```





4.1.2 Detallando las variables estadísticas

Ahora veremos a detalle lo que ocurre con todas y cada una de las variables estadísticas.

```
In [ ]: data['bill_length_mm'].describe()
```



```
Out[ ]: count    333.000000
        mean     43.992793
        std      5.468668
        min     32.100000
        25%     39.500000
        50%     44.500000
        75%     48.600000
        max     59.600000
        Name: bill_length_mm, dtype: float64
```

```
In [ ]: data['bill_depth_mm'].describe()
```

```
Out[ ]: count    333.000000
        mean     17.164865
        std      1.969235
        min     13.100000
        25%     15.600000
        50%     17.300000
        75%     18.700000
        max     21.500000
        Name: bill_depth_mm, dtype: float64
```

```
In [ ]: data['flipper_length_mm'].describe()
```

```
Out[ ]: count    333.000000
        mean     200.966967
        std      14.015765
        min     172.000000
        25%     190.000000
        50%     197.000000
        75%     213.000000
        max     231.000000
        Name: flipper_length_mm, dtype: float64
```

```
In [ ]: data['body_mass_g'].describe()
```

```
Out[ ]: count    333.000000
        mean     4207.057057
        std      805.215802
        min     2700.000000
        25%     3550.000000
        50%     4050.000000
        75%     4775.000000
        max     6300.000000
        Name: body_mass_g, dtype: float64
```

```
In [ ]: data['year'].describe()
```

```
Out[ ]: count      333.000000  
        mean       2008.042042  
        std         0.812944  
        min        2007.000000  
        25%        2007.000000  
        50%        2008.000000  
        75%        2009.000000  
        max        2009.000000  
        Name: year, dtype: float64
```

4.2 Análisis univariado

4.3 Análisis bivariado

Conclusiones