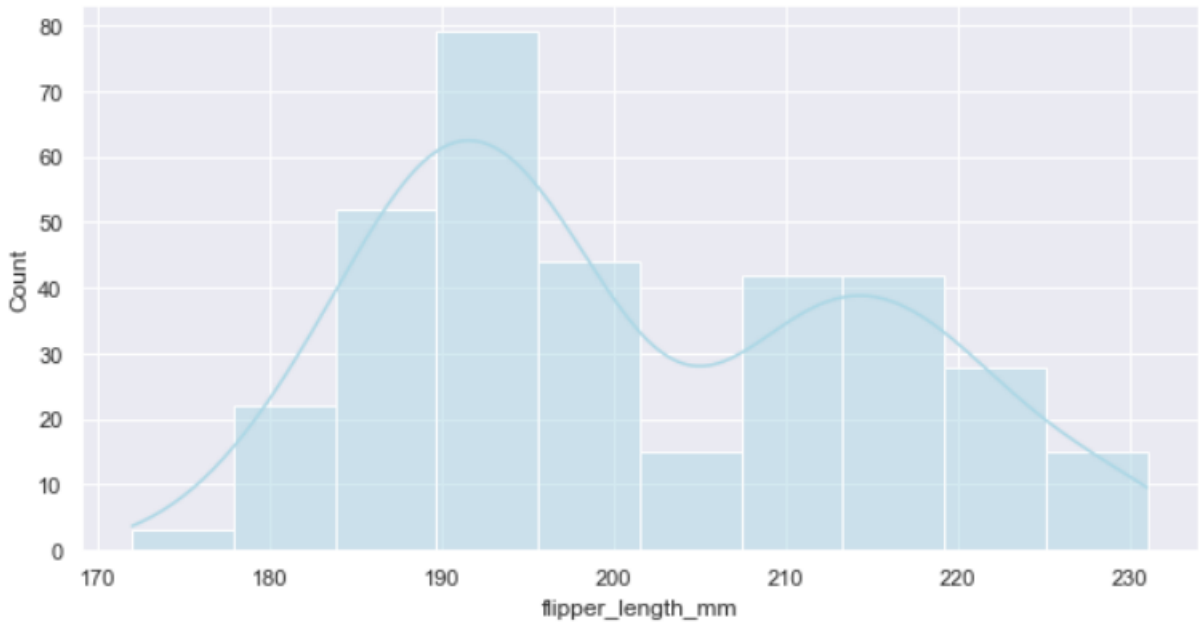
Introducing Palmer Penguins

The iris dataset is one that is commonly used for visualisation and pattern recognition in Data Science. Since recently discovering the author's link with eugenics, there has been an effort to source other datasets for use.

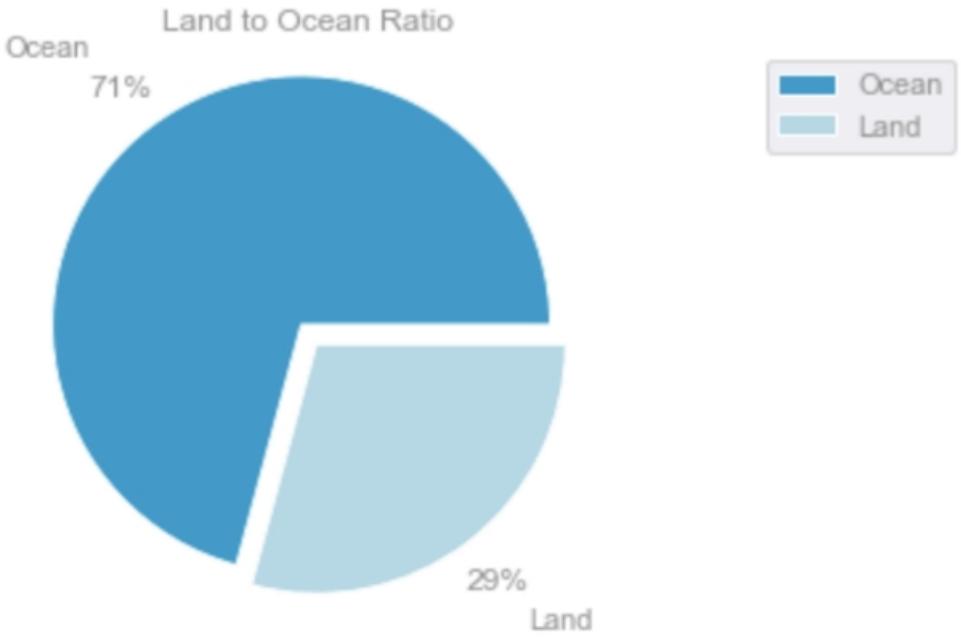
The above histogram display the disribution of the penguins flipper_length in mm.here, the bin values can be confirmed using the output.

```
import seaborn as sns
penguins = sns.load_dataset('penguins')
sns.histplot(data=penguins['flipper_length_mm'], kde=True);
```



Pie chart :
a pie chart is a visualization of univariate data tha depicts the data in a circular diagram. Each pie chart slice corresponds to a relative proportion of the category versus the entire group.

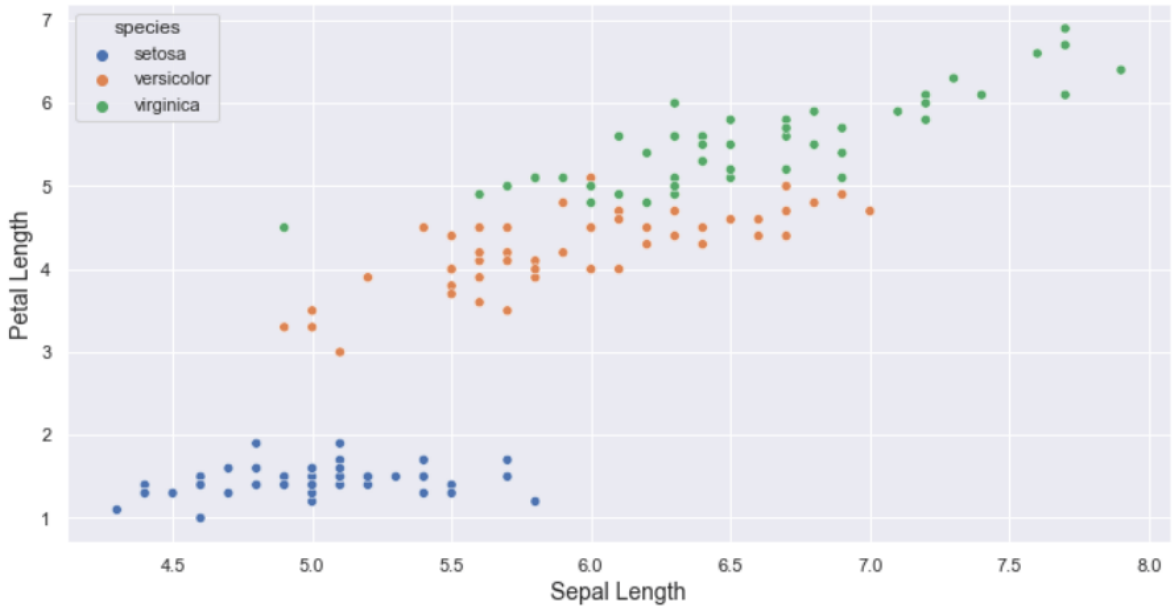
```
import seaborn as sns
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
labels = ['Ocean', 'Land']
color_palette_list = ['#009ACD', '#ADD8E6']

percentages = [70.8, 29.2]
explode=(0.1,0)
ax.pie(percentages, explode=explode, labels=labels,
       colors=color_palette_list[0:2], autopct='%1.0f%%',
       shadow=False, startangle=0,
       pctdistance=1.2,labeldistance=1.4)


ax.axis('equal')
ax.set_title("Land to Ocean Ratio")
ax.legend(bbox_to_anchor=(1, 1));
```



Land to Ocean area Pie Chart

Scatter plot:

```
iris = sns.load_dataset('iris')
sns.scatterplot(data=iris, x='sepal_length', y='petal_length', hue='species')
plt.xlabel('Sepal Length')
plt.ylabel('Petal Length')
plt.show()
```

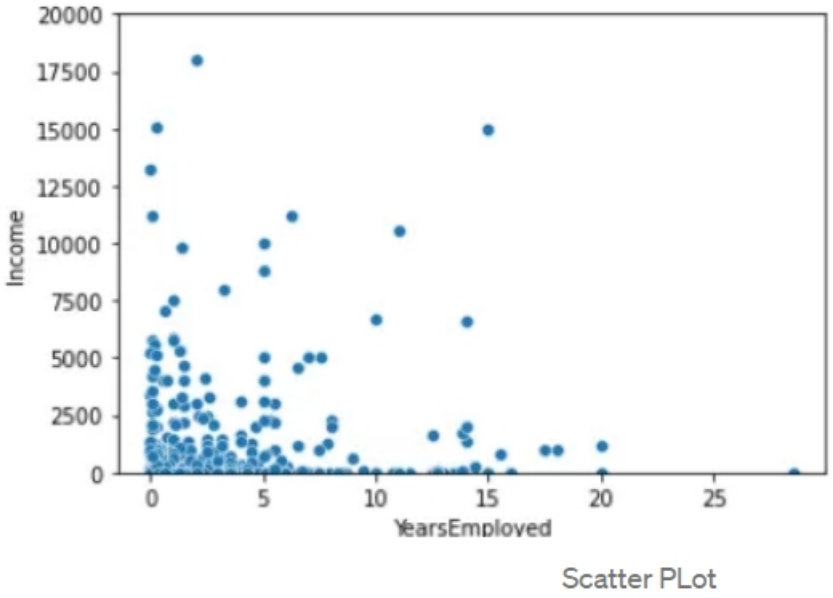


Though in this dataset, we don't see any strong correlation between any two continuous variables, in some datasets, continuous variables could be strongly correlated and the values of one might depend on others.

We can also draw line plots and scatterplots to see a relation between the two continuous variables

```
sns.scatterplot(card_approval_data.YearsEmployed, card_approval_data.Income)
plt.ylim(0,20000)
```

```
(0.0, 20000.0)
```



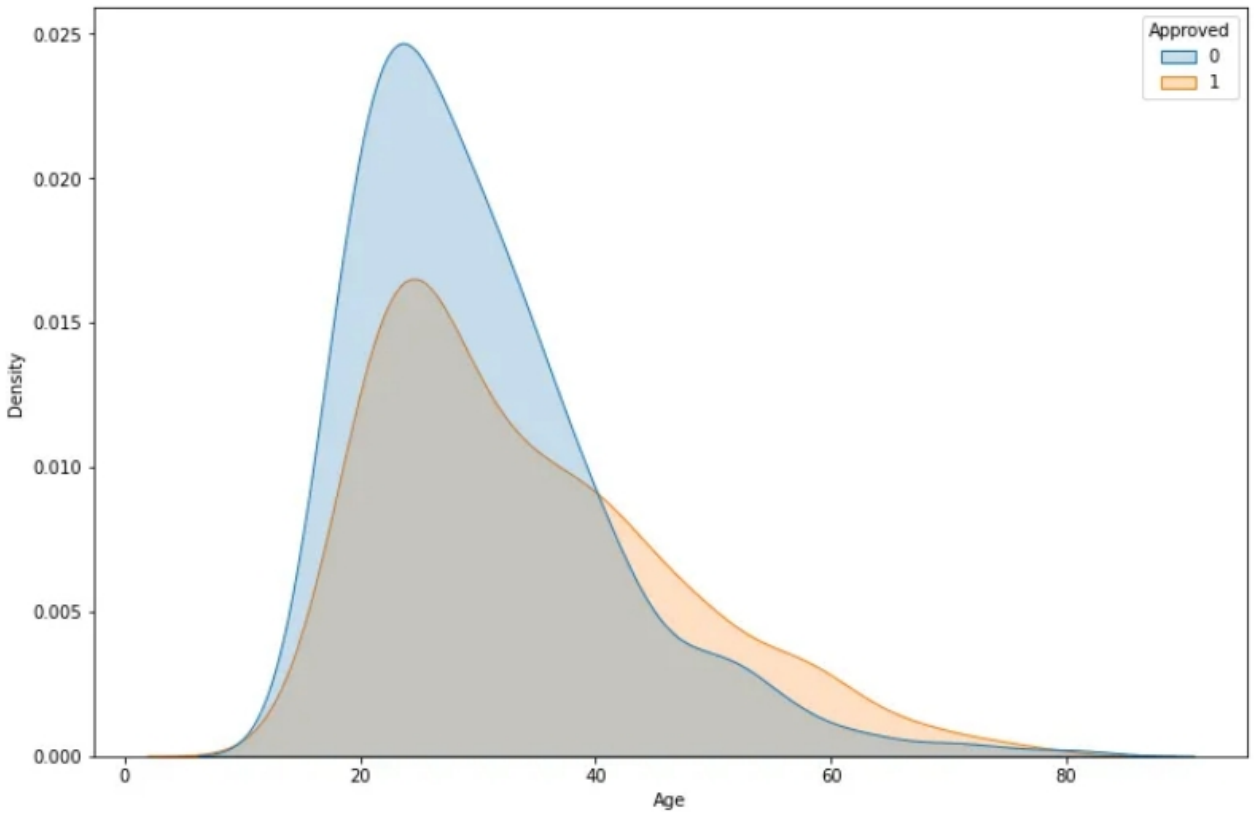
Scatter PLot

KDE Plots with Hue: A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.

We will plot KDE plots of continius variables with hue='Approved'

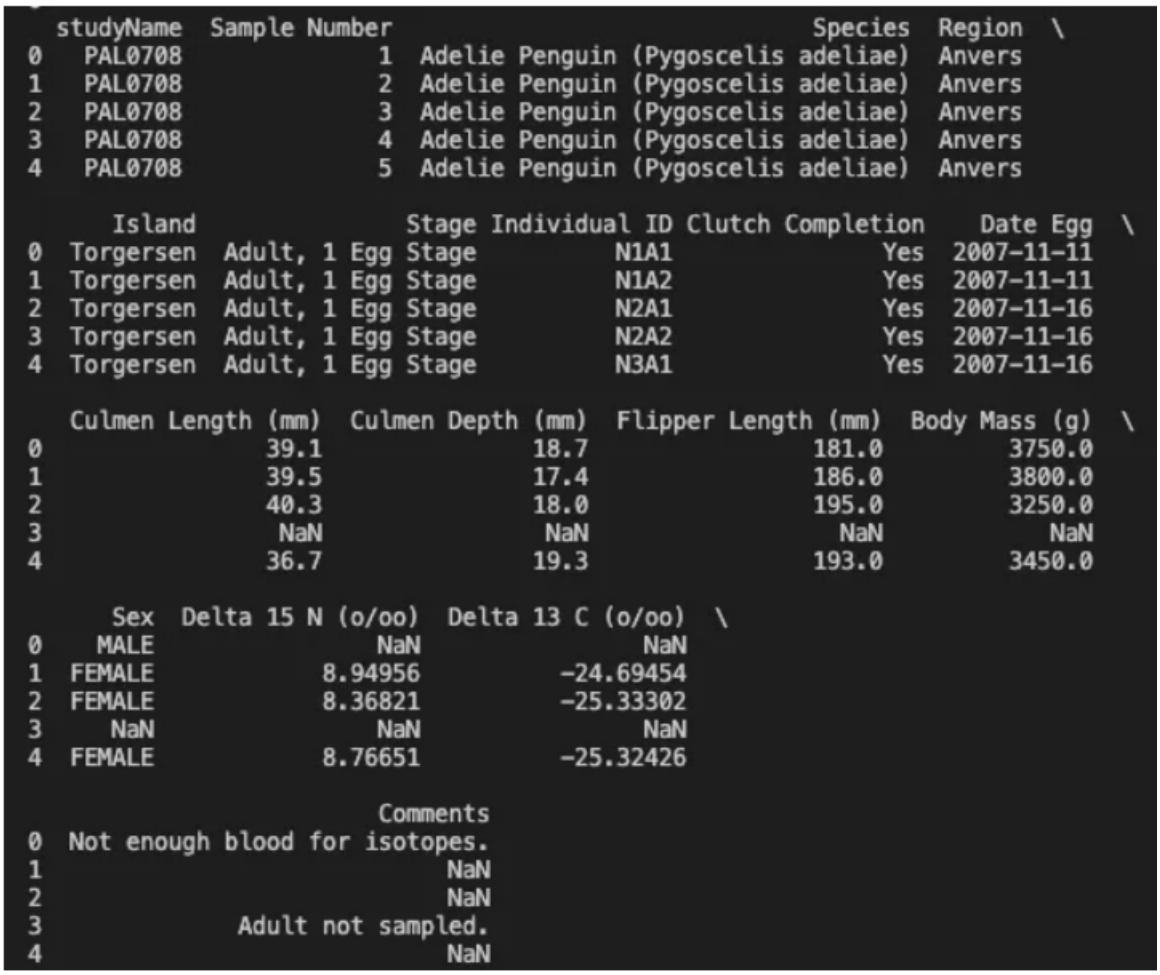
```
plt.figure(figsize=(12,8))
sns.kdeplot(data=card_approval_data, x='Age',hue='Approved',fill=True)
```

```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



KDE Plot for 'Age' Column wiht Code to plot it

```
df.to_csv('output.csv', compression='gzip')
pd.set_option('display.max_columns', None)
print(df.head())
```



	studyName	Sample Number	Species	Region	\
0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	
1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	
2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	
3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	
4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	

	Island	Stage	Individual ID	Clutch Completion	Date Egg	\
0	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	2007-11-11	
1	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	2007-11-11	
2	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	2007-11-16	
3	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	2007-11-16	
4	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	2007-11-16	

	Culmen Length (mm)	Culmen Depth (mm)	Flipper Length (mm)	Body Mass (g)	\
0	39.1	18.7	181.0	3750.0	
1	39.5	17.4	186.0	3800.0	
2	40.3	18.0	195.0	3250.0	
3	NaN	NaN	NaN	NaN	
4	36.7	19.3	193.0	3450.0	

	Sex	Delta 15 N (o/oo)	Delta 13 C (o/oo)	\
0	MALE	NaN	NaN	
1	FEMALE	8.94956	-24.69454	
2	FEMALE	8.36821	-25.33302	
3	NaN	NaN	NaN	
4	FEMALE	8.76651	-25.32426	

	Comments
0	Not enough blood for isotopes.
1	NaN
2	NaN
3	Adult not sampled.
4	NaN

Output of df.head() after setting the max columns to be None

In this article, we will be taking a look at how to handle missing data in Python using the Pandas library.

```
df = pd.read_csv('https://raw.githubusercontent.com/dataprofessor/data/master/
penguins_cleaned.csv')
df
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181	3750	male
1	Adelie	Torgersen	39.5	17.4	186	3800	female
2	Adelie	Torgersen	40.3	18.0	195	3250	female
3	Adelie	Torgersen	36.7	19.3	193	3450	female
4	Adelie	Torgersen	39.3	20.6	190	3650	male
...	...	...	...	...	...	...	...
328	Chinstrap	Dream	55.8	19.8	207	4000	male
329	Chinstrap	Dream	43.5	18.1	202	3400	female
330	Chinstrap	Dream	49.6	18.2	193	3775	male
331	Chinstrap	Dream	50.8	19.0	210	4100	male
332	Chinstrap	Dream	50.2	18.7	198	3775	female

333 rows × 7 columns

Add missing data

Because our dataset is already cleaned, therefore there is no missing data. Thus, let's add some missing values to it.

```
df.iloc[0, 0] = pd.NA
df.iloc[5, 1] = pd.NA
df.iloc[10, 2] = pd.NA
df.iloc[15, 3] = pd.NA
df.iloc[20, 4] = pd.NA
df.iloc[25, 5] = pd.NA
df.iloc[30, 6] = pd.NA

df.isna()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	True	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
328	False	False	False	False	False	False	False
329	False	False	False	False	False	False	False
330	False	False	False	False	False	False	False
331	False	False	False	False	False	False	False
332	False	False	False	False	False	False	False

333 rows × 7 columns

```
df[df.isnull().any(axis=1)]
```

where `axis=1` refers to the columns while the `any()` function literally tells it to display any rows consisting of at least 1 missing value.

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	<NA>	Torgersen	39.1	18.7	181	3750	male
5	Adelie	<NA>	38.9	17.8	181	3625	female
10	Adelie	Torgersen	<NA>	17.8	185	3700	female
15	Adelie	Biscoe	37.8	<NA>	174	3400	female
20	Adelie	Biscoe	35.3	18.9	<NA>	3800	female
25	Adelie	Dream	39.5	16.7	178	<NA>	female
30	Adelie	Dream	39.2	21.1	196	4150	<NA>

Check for non-missing data

We can also check for non-missing data and there are 2 functions for this that includes `notna()` and `notnull()`. It should be noted that both of these functions will produce the same output and therefore you can use either one.

```
df.notna()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	False	True	True	True	True	True	True
1	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True
...	...	...	...	...	...	...	...
328	True	True	True	True	True	True	True
329	True	True	True	True	True	True	True
330	True	True	True	True	True	True	True
331	True	True	True	True	True	True	True
332	True	True	True	True	True	True	True

333 rows × 7 columns

Replace missing value with 0

To fill in missing values with a numerical value of 0 (i.e. the value of 0 is an arbitrary value and this can be any other value of your choice), type the following:

```
df.fillna(0)
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	0	Torgersen	39.1	18.7	181	3750	male
1	Adelie	Torgersen	39.5	17.4	186	3800	female
2	Adelie	Torgersen	40.3	18.0	195	3250	female
3	Adelie	Torgersen	36.7	19.3	193	3450	female
4	Adelie	Torgersen	39.3	20.6	190	3650	male
...	...	...	...	...	...	...	...
328	Chinstrap	Dream	55.8	19.8	207	4000	male
329	Chinstrap	Dream	43.5	18.1	202	3400	female
330	Chinstrap	Dream	49.6	18.2	193	3775	male
331	Chinstrap	Dream	50.8	19.0	210	4100	male
332	Chinstrap	Dream	50.2	18.7	198	3775	female

333 rows × 7 columns

Fill missing value with the mean value

Instead, let's say that you want to fill in the missing value with the mean value for each column, you can type the following:

```
df.fillna(df.mean())
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	<NA>	Torgersen	39.1	18.7	181	3750	male
1	Adelie	Torgersen	39.5	17.4	186	3800	female
2	Adelie	Torgersen	40.3	18	195	3250	female
3	Adelie	Torgersen	36.7	19.3	193	3450	female
4	Adelie	Torgersen	39.3	20.6	190	3650	male
...	...	...	...	...	...	...	...
328	Chinstrap	Dream	55.8	19.8	207	4000	male
329	Chinstrap	Dream	43.5	18.1	202	3400	female
330	Chinstrap	Dream	49.6	18.2	193	3775	male
331	Chinstrap	Dream	50.8	19	210	4100	male
332	Chinstrap	Dream	50.2	18.7	198	3775	female

333 rows × 7 columns