title: “EDA_Phyton” author: “Andrea B” date: “2023-11-16” output: pdf_document: pdf_print: paged —

Pokémon con Estadísticas

Este conjunto de datos incluye 721 Pokémon, incluido su número, nombre, primer y segundo tipo, y estadísticas básicas: HP, Ataque, Defensa, Ataque Especial, Defensa Especial y Velocidad. Ha sido de gran utilidad para enseñar estadística a los niños.

Para la visualización del conjunto de datos vamos a cargar la siguiente libreria:

```
library(readr)
```

Importar la base de datos del archivo Pokemon

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
Pokemon=pd.read_csv(r"C:\Users\Andrea\Desktop\Pokemon.csv")

print(Pokemon)
```

```
##        #                   Name   Type 1  ...  Speed  Generation  Legendary
## 0      1              Bulbasaur    Grass  ...     45           1      False
## 1      2                Ivysaur    Grass  ...     60           1      False
## 2      3               Venusaur    Grass  ...     80           1      False
## 3      3  VenusaurMega Venusaur    Grass  ...     80           1      False
## 4      4             Charmander     Fire  ...     65           1      False
## ..   ...                    ...      ...  ...    ...         ...        ...
## 795  719                Diancie     Rock  ...     50           6       True
## 796  719    DiancieMega Diancie     Rock  ...    110           6       True
## 797  720    HoopaHoopa Confined  Psychic  ...     70           6       True
## 798  720     HoopaHoopa Unbound  Psychic  ...     80           6       True
## 799  721              Volcanion     Fire  ...     70           6       True
##
## [800 rows x 13 columns]
```

¿Cuál es la Distribución de la Generación?

A continuación se realizará el conteo del número de pokemon que se encuentran en cada generación.

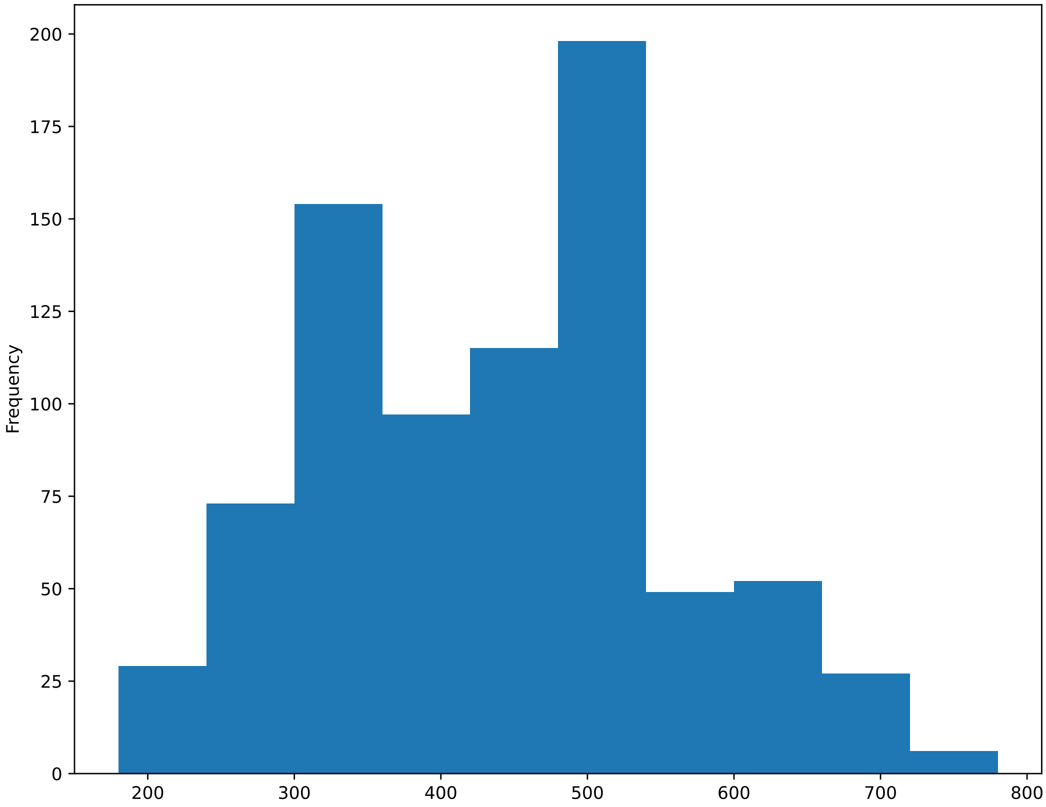
```
Pokemon.value_counts('Type 1')
```

```
## Type 1
## Water       112
## Normal       98
## Grass        70
## Bug          69
## Psychic      57
## Fire         52
## Electric     44
## Rock         44
## Ghost        32
## Ground       32
## Dragon       32
## Dark         31
## Poison       28
## Fighting     27
## Steel        27
## Ice          24
## Fairy        17
## Flying        4
## Name: count, dtype: int64
```

```
Pokemon.describe()
```

```
##                 #      Total          HP  ...     Sp. Def       Speed  Generation
## count  800.000000  800.00000  800.000000  ...  800.000000  800.000000   800.00000
## mean   362.813750  435.10250   69.258750  ...   71.902500   68.277500     3.32375
## std    208.343798  119.96304   25.534669  ...   27.828916   29.060474     1.66129
## min      1.000000  180.00000    1.000000  ...   20.000000    5.000000     1.00000
## 25%    184.750000  330.00000   50.000000  ...   50.000000   45.000000     2.00000
## 50%    364.500000  450.00000   65.000000  ...   70.000000   65.000000     3.00000
## 75%    539.250000  515.00000   80.000000  ...   90.000000   90.000000     5.00000
## max    721.000000  780.00000  255.000000  ...  230.000000  180.000000     6.00000
##
## [8 rows x 9 columns]
```

```
Pokemon["Total"].plot(kind='hist',figsize=(10,8))
```



¿En que generación es mas probable encontrar un pokemon no lengendario con un ataque alto?.

Dentro del analisis realizado se relaciona el codigo a continuacion del conteo de los pokemon legendarios arrojando los siguientes datos:

```
Pokemon["Attack"].isin(["True", "False"])
```

```
## 0      False
## 1      False
## 2      False
## 3      False
## 4      False
##         ...
## 795    False
## 796    False
## 797    False
## 798    False
## 799    False
## Name: Attack, Length: 800, dtype: bool
```

```
Pokemon.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 800 entries, 0 to 799
## Data columns (total 13 columns):
##  #   Column      Non-Null Count  Dtype
## ---  ------      --------------  -----
##  0   #           800 non-null    int64
##  1   Name        800 non-null    object
##  2   Type 1      800 non-null    object
##  3   Type 2      414 non-null    object
##  4   Total       800 non-null    int64
##  5   HP          800 non-null    int64
##  6   Attack      800 non-null    int64
##  7   Defense     800 non-null    int64
##  8   Sp. Atk     800 non-null    int64
##  9   Sp. Def     800 non-null    int64
##  10  Speed       800 non-null    int64
##  11  Generation  800 non-null    int64
##  12  Legendary   800 non-null    bool
## dtypes: bool(1), int64(9), object(3)
## memory usage: 75.9+ KB
```

```
Pokemon["Attack"].sum()
```

```
## 63201
```

```
Pokemon.loc[Pokemon["Generation"]].shape
```

```
## (800, 13)
```

Pokemon Legendario y No Legendario

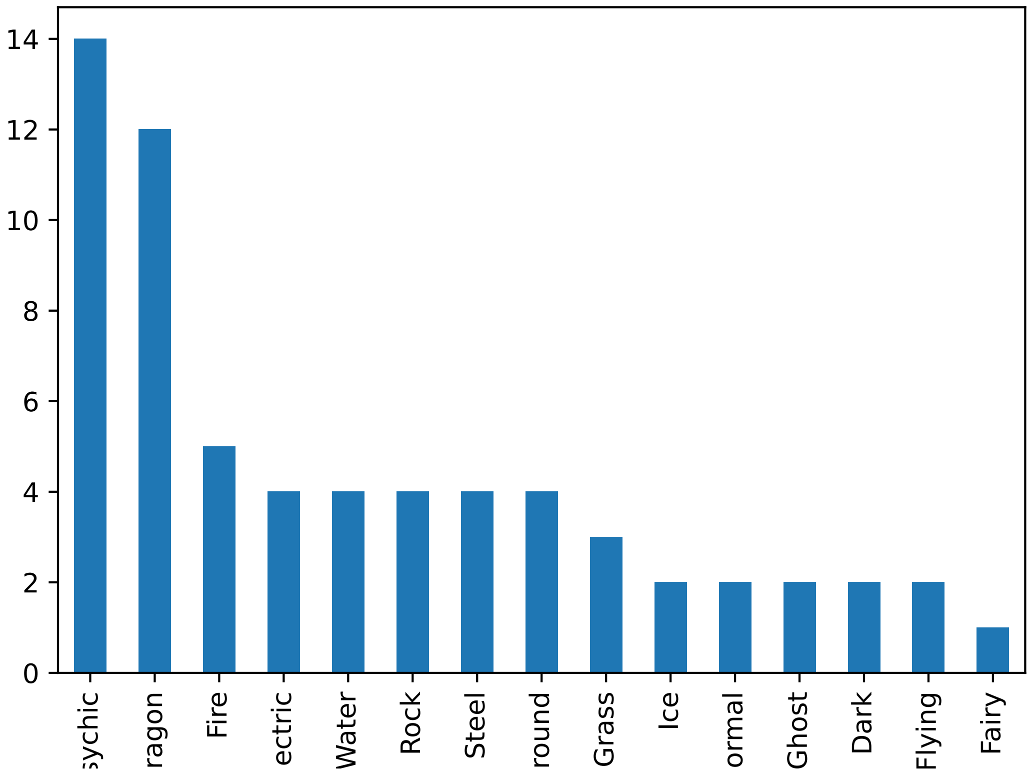
Dentro del analisis de datos de Pokemon vamos a realizar la comparación del tipo más común de pokemon legendario.

Pokemon Legendario

```
Pokemon.loc[Pokemon["Legendary"],"Type 1"].value_counts()
```

```
## Type 1
## Psychic     14
## Dragon      12
## Fire         5
## Electric     4
## Water        4
## Rock         4
## Steel        4
## Ground       4
## Grass        3
## Ice          2
## Normal       2
## Ghost        2
## Dark         2
## Flying       2
## Fairy        1
## Name: count, dtype: int64
```

```
Pokemon.loc[Pokemon["Legendary"],"Type 1"].value_counts().plot(kind="bar")
```



Pokemon No Legendario

```
ordinary = Pokemon[Pokemon["Legendary"] == False].reset_index(drop=True)
print(ordinary.shape)
```

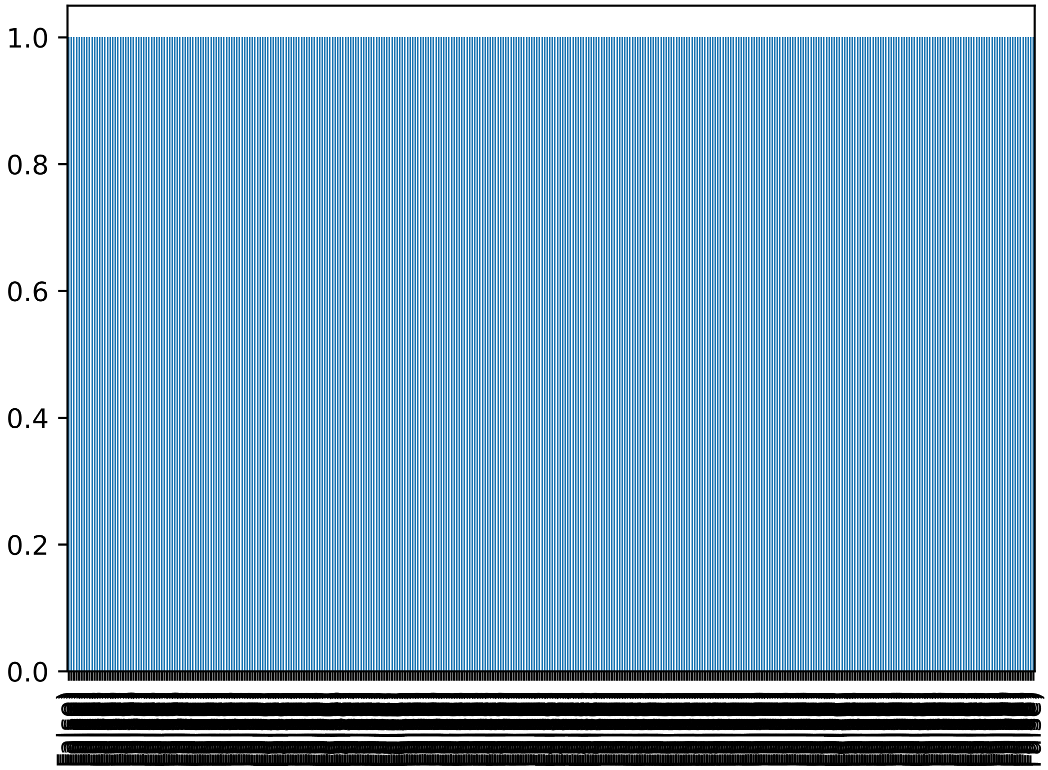
```
## (735, 13)
```

```
ordinary.head()
```

```
##    #                  Name Type 1  ... Speed  Generation  Legendary
## 0  1             Bulbasaur  Grass  ...    45           1      False
## 1  2               Ivysaur  Grass  ...    60           1      False
## 2  3              Venusaur  Grass  ...    80           1      False
## 3  3  VenusaurMega Venusaur  Grass  ...    80           1      False
## 4  4            Charmander   Fire  ...    65           1      False
##
## [5 rows x 13 columns]
```

```
ordinary = Pokemon[Pokemon["Legendary"] == False].value_counts()
```

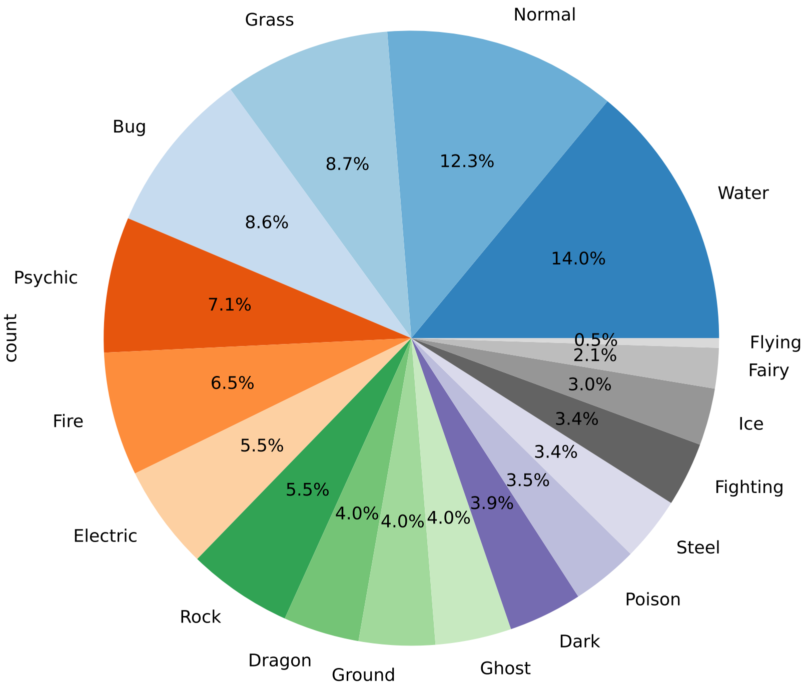
```
ordinary = Pokemon[Pokemon["Legendary"] == False].value_counts().plot(kind="bar")
```



Analisis Final de Pokemon

Durante el desarrollo de este proyecto se pudo visualizar como estan cada uno catalogados cada uno de los pokemons a continuación se muestra un grafico de acuerdo a su categoria.

```
Pokemon["Type 1"].value_counts().plot(kind="pie",autopct="%1.1f%%",cmap='tab20c',figsize=(10,8))
```

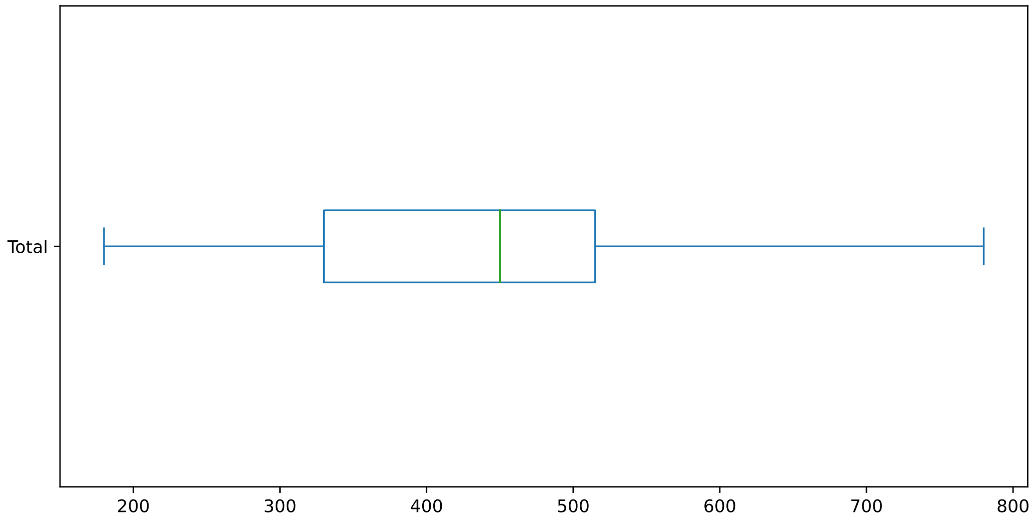


Su distribución total.

```
Pokemon.head()
```

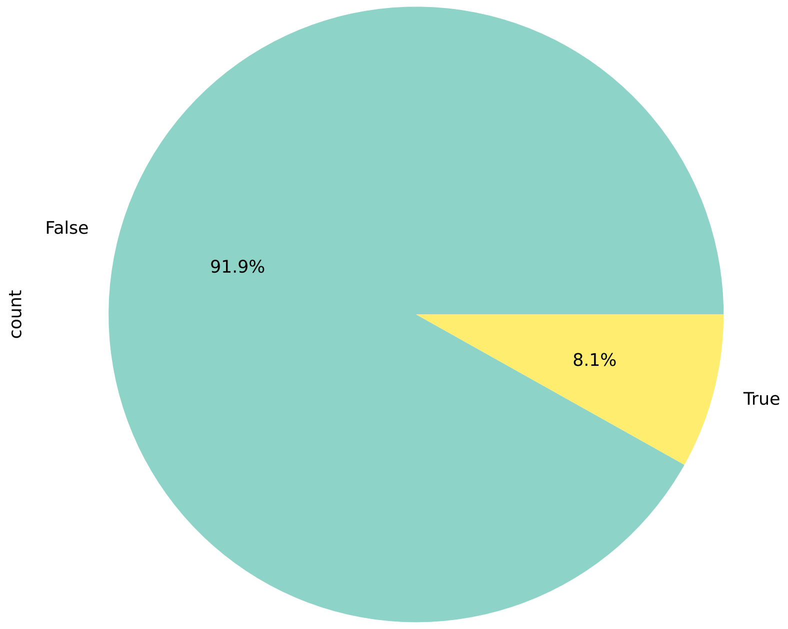
```
##    #                   Name Type 1  ... Speed  Generation  Legendary
## 0  1              Bulbasaur  Grass  ...    45           1      False
## 1  2                Ivysaur  Grass  ...    60           1      False
## 2  3               Venusaur  Grass  ...    80           1      False
## 3  3  VenusaurMega Venusaur  Grass  ...    80           1      False
## 4  4             Charmander   Fire  ...    65           1      False
##
## [5 rows x 13 columns]
```

```
Pokemon["Total"].plot(kind='box',vert=False,figsize=(10,5))
```



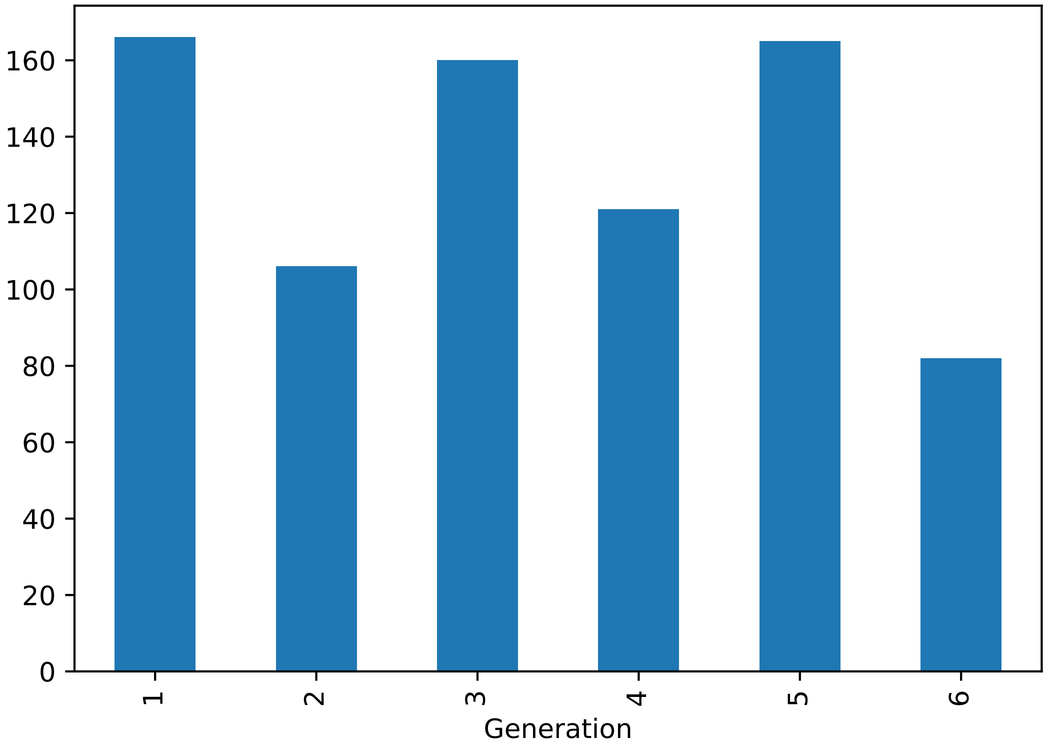
Distribución de pokemon legendarios.

```
Pokemon["Legendary"].value_counts().plot(kind="pie",autopct="%1.1f%%",cmap="Set3",figsize=(10,8))
```



Podemos visualizar cual es el pokemon mas poderoso de las 3 primeras generaciones, del tipo agua.

```
Pokemon["Generation"].value_counts(sort=False).plot(kind="bar")
```



```
(
    (Pokemon["Generation"]==1)|
    (Pokemon["Generation"]==2) |
    (Pokemon["Generation"]==3)
).sum()
```

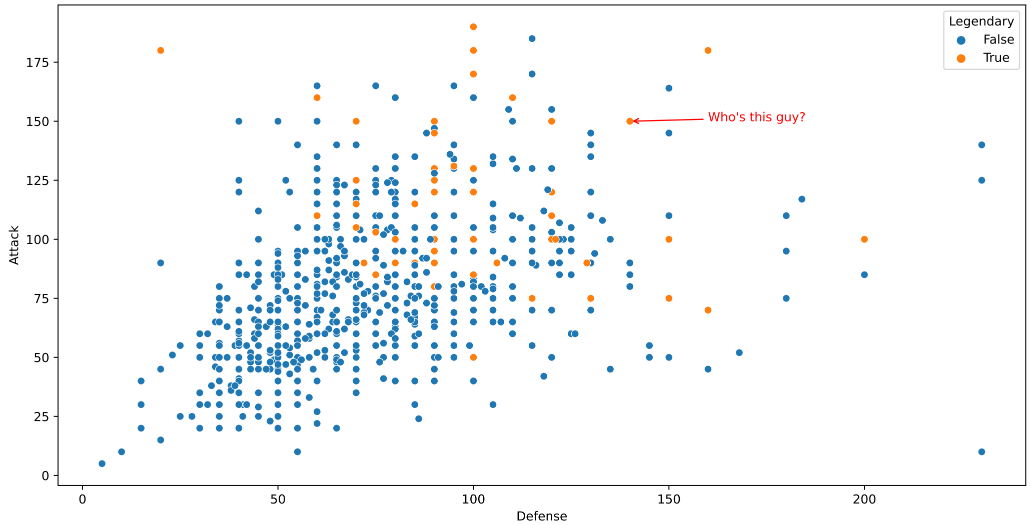
```
## 432
```

```
Pokemon.loc[
    (Pokemon["Type 1"]=="Water")&
    Pokemon["Generation"].isin([1,2,3])
].sort_values(by="Total",ascending=False).head()
```

```
##        #                    Name Type 1  ... Speed  Generation  Legendary
## 422  382      KyogrePrimal Kyogre  Water  ...    90           3       True
## 421  382                   Kyogre  Water  ...    90           3       True
## 141  130    GyaradosMega Gyarados  Water  ...    81           1      False
## 283  260    SwampertMega Swampert  Water  ...    70           3      False
## 12     9  BlastoiseMega Blastoise  Water  ...    78           1      False
##
## [5 rows x 13 columns]
```

El Pokémon legendario ultrapoderoso se muestra a continuación.

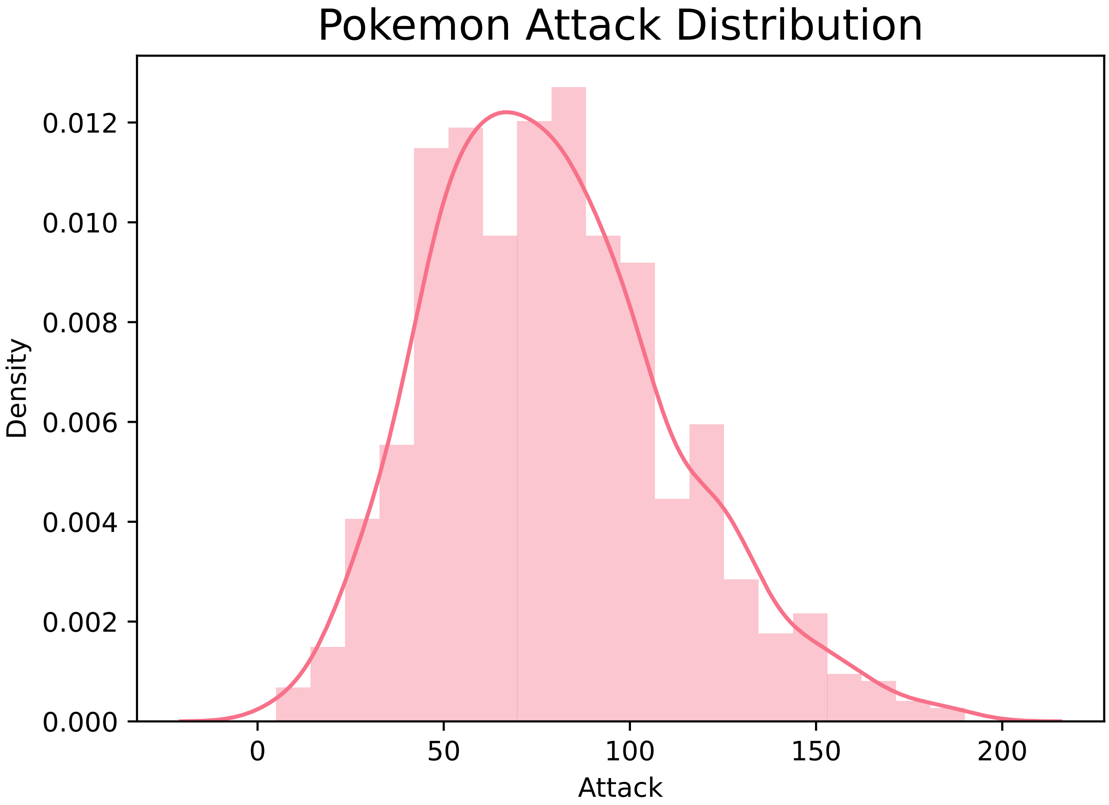
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(14, 7))
sns.scatterplot(data=Pokemon, x="Defense", y="Attack", hue='Legendary', ax=ax)
ax.annotate(
    "Who's this guy?", xy=(140, 150), xytext=(160, 150), color='red',
    arrowprops=dict(arrowstyle="->", color='red')
)
```



```
sns.set_palette("husl", 8)
ax = sns.distplot(Pokemon['Attack'])
```

```
## <string>:1: UserWarning:
##
## `distplot` is a deprecated function and will be removed in seaborn v0.14.0.
##
## Please adapt your code to use either `displot` (a figure-level function with
## similar flexibility) or `histplot` (an axes-level function for histograms).
##
## For a guide to updating your code to use the new functions, please see
## https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
##
## C:\Users\Andrea\AppData\Local\Programs\Python\PYTHON~1\Lib\site-packages\seaborn\_oldcore.py:1498: Fu
##   if pd.api.types.is_categorical_dtype(vector):
## C:\Users\Andrea\AppData\Local\Programs\Python\PYTHON~1\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
##   with pd.option_context('mode.use_inf_as_na', True):
```

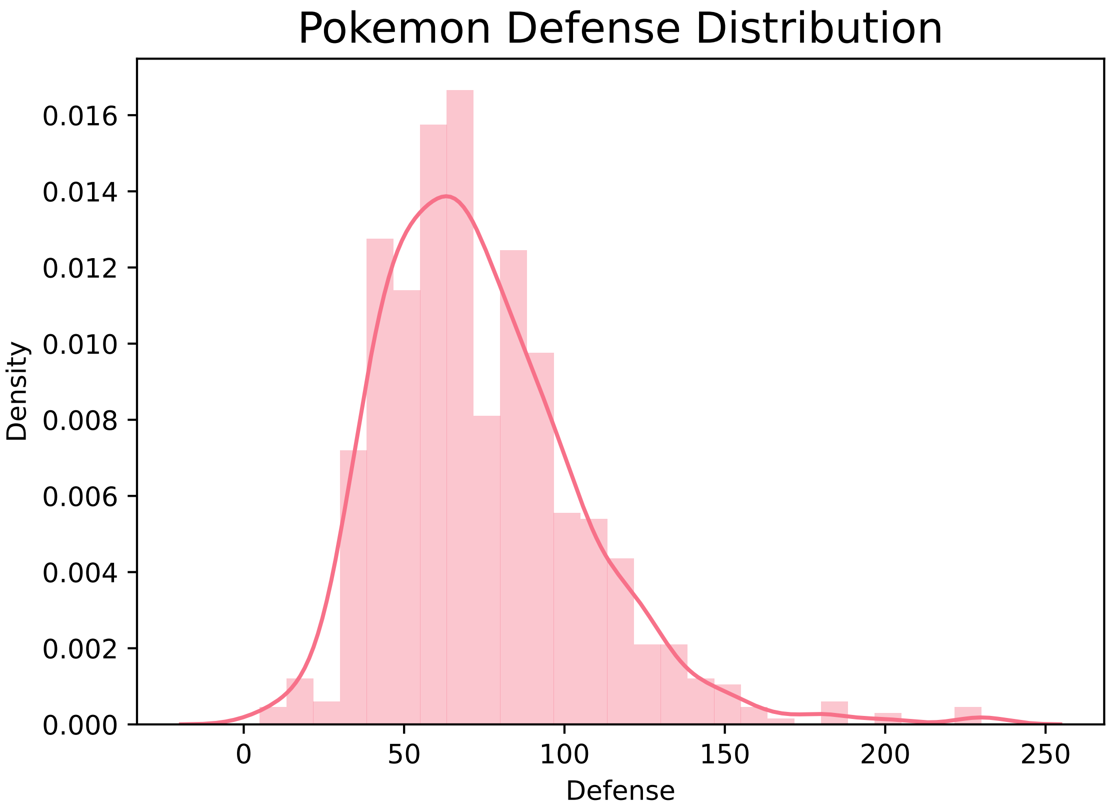
```
ax.set_title("Pokemon Attack Distribution", fontdict={'fontsize': 16})
```



```
sns.set_palette("husl", 8)
ax = sns.distplot(Pokemon['Defense'])
```

```
## <string>:1: UserWarning:
##
## `distplot` is a deprecated function and will be removed in seaborn v0.14.0.
##
## Please adapt your code to use either `displot` (a figure-level function with
## similar flexibility) or `histplot` (an axes-level function for histograms).
##
## For a guide to updating your code to use the new functions, please see
## https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
##
## C:\Users\Andrea\AppData\Local\Programs\Python\PYTHON~1\Lib\site-packages\seaborn\_oldcore.py:1498: Fu
##   if pd.api.types.is_categorical_dtype(vector):
## C:\Users\Andrea\AppData\Local\Programs\Python\PYTHON~1\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
##   with pd.option_context('mode.use_inf_as_na', True):
```

```
ax.set_title("Pokemon Defense Distribution", fontdict={'fontsize': 16})
```



```
mean_attack_generation = Pokemon.groupby("Generation")["Attack"].mean().sort_values()
print(mean_attack_generation)
```

```
## Generation
## 2    72.028302
## 6    75.804878
## 1    76.638554
## 3    81.625000
## 5    82.066667
## 4    82.867769
## Name: Attack, dtype: float64
```

```
mean_defense_generation = Pokemon.groupby("Generation")["Defense"].mean().sort_values()
print(mean_defense_generation)
```

```
## Generation
## 1    70.861446
## 5    72.327273
## 2    73.386792
## 3    74.100000
## 6    76.682927
## 4    78.132231
## Name: Defense, dtype: float64
```