Project: Querying and Filtering Pokemon data

This project will help you practice your pandas querying and filtering skills. Let's begin!

Photo by Mikel (https://unsplash.com/@mykelgran?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on Unsplash (https://unsplash.com/s/photos/pokemon?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText).

Task 0 - Setup

There isn't much to do here, we'll provide the required imports and the read the pokemon CSV we'll be working with.

In [109]:
```
import numpy as np
import pandas as pd

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

In [110]:
```
df = pd.read_csv("pokemon.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False
4	5	Charmeleon	Fire	NaN	405	58	64	58	80	65	80	1	False

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 721 entries, 0 to 720
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   #           721 non-null    int64
 1   Name        721 non-null    object
 2   Type 1      721 non-null    object
 3   Type 2      359 non-null    object
 4   Total       721 non-null    int64
 5   HP          721 non-null    int64
 6   Attack      721 non-null    int64
 7   Defense     721 non-null    int64
 8   Sp. Atk     721 non-null    int64
 9   Sp. Def     721 non-null    int64
 10  Speed       721 non-null    int64
 11  Generation  721 non-null    int64
 12  Legendary   721 non-null    bool
dtypes: bool(1), int64(9), object(3)
memory usage: 68.4+ KB
```

In [5]:
```
df.describe()
```

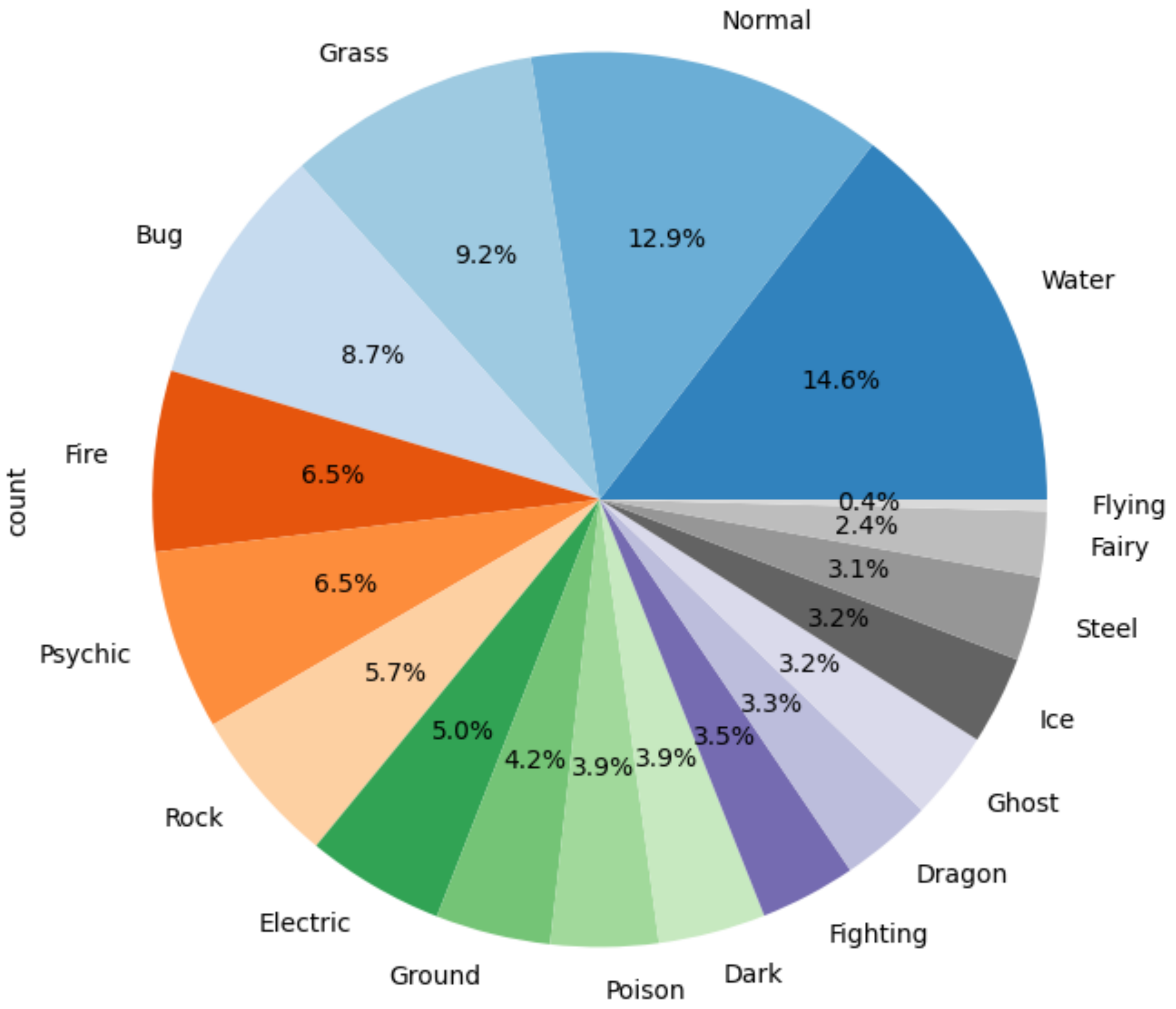
Out[5]:

	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
count	721.00000	721.000000	721.000000	721.000000	721.000000	721.000000	721.000000	721.000000	721.000000
mean	361.00000	417.945908	68.380028	75.124827	70.697642	68.848821	69.180305	65.714286	3.323162
std	208.27906	109.663671	25.848272	29.070335	29.194941	28.898590	26.899364	27.277920	1.669873
min	1.00000	180.000000	1.000000	5.000000	5.000000	10.000000	20.000000	5.000000	1.000000
25%	181.00000	320.000000	50.000000	54.000000	50.000000	45.000000	50.000000	45.000000	2.000000
50%	361.00000	424.000000	65.000000	75.000000	65.000000	65.000000	65.000000	65.000000	3.000000
75%	541.00000	499.000000	80.000000	95.000000	85.000000	90.000000	85.000000	85.000000	5.000000
max	721.00000	720.000000	255.000000	165.000000	230.000000	154.000000	230.000000	160.000000	6.000000

Distribution of Pokemon Types:

```
In [6]: df['Type 1'].value_counts().plot(kind='pie', autopct='%1.1f%%', cmap='tab20c', figsize=(10, 8))
```

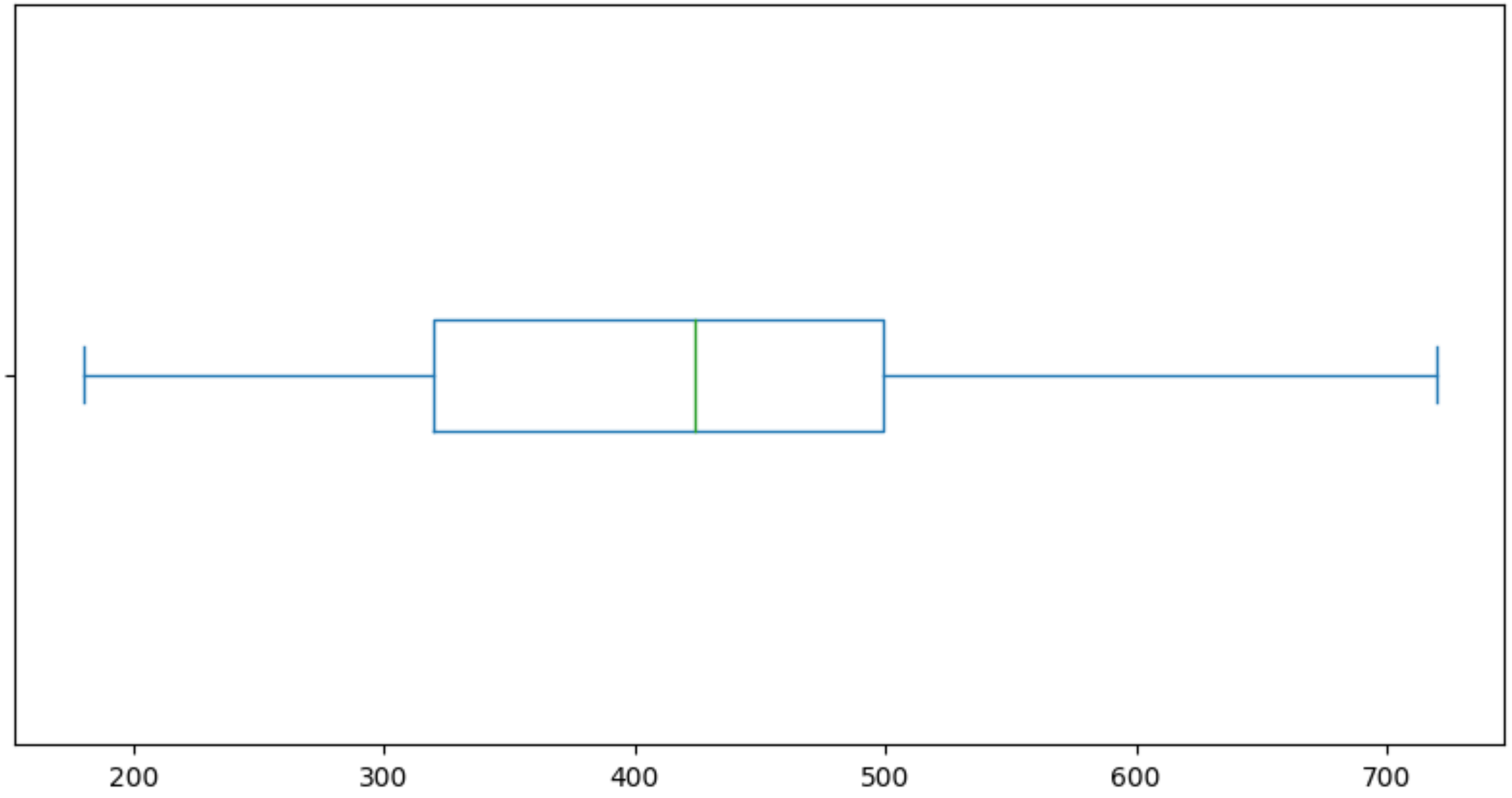
```
Out[6]: <Axes: ylabel='count'>
```

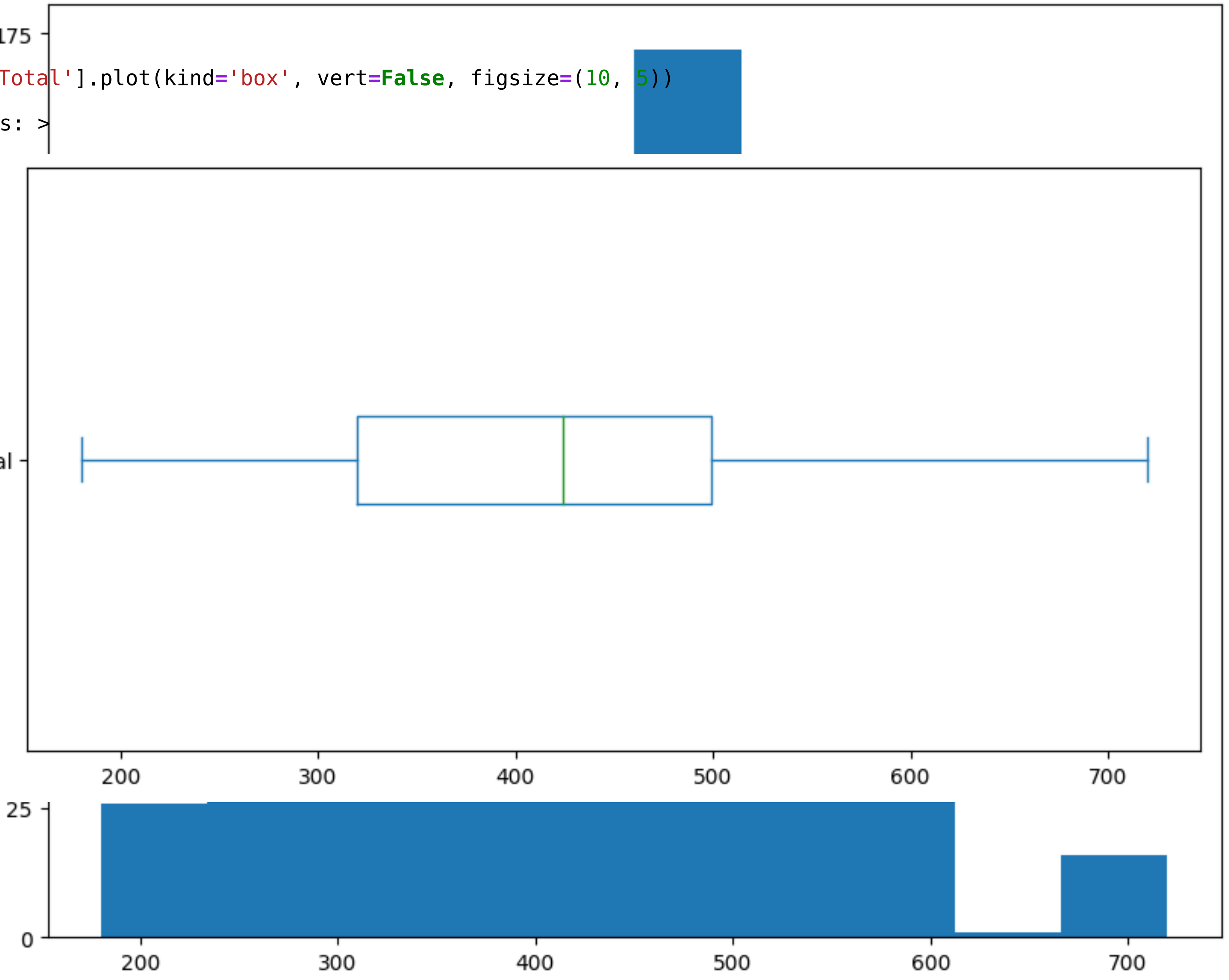

count
Water 14.6%
Normal 12.9%
Grass 9.2%
Bug 8.7%
Fire 6.5%
Psychic 6.5%
Rock 5.7%
Electric 5.0%
Ground 4.2%
Poison 3.9%
Dark 3.9%
Fighting 3.5%
Dragon 3.3%
Ghost 3.2%
Ice 3.2%
Steel 3.1%
Fairy 2.4%
Flying 0.4%

```
In [7]: df['Total'].plot(kind='hist', figsize=(10, 8))
```

```
Out[7]: <Axes: ylabel='Frequency'>
```

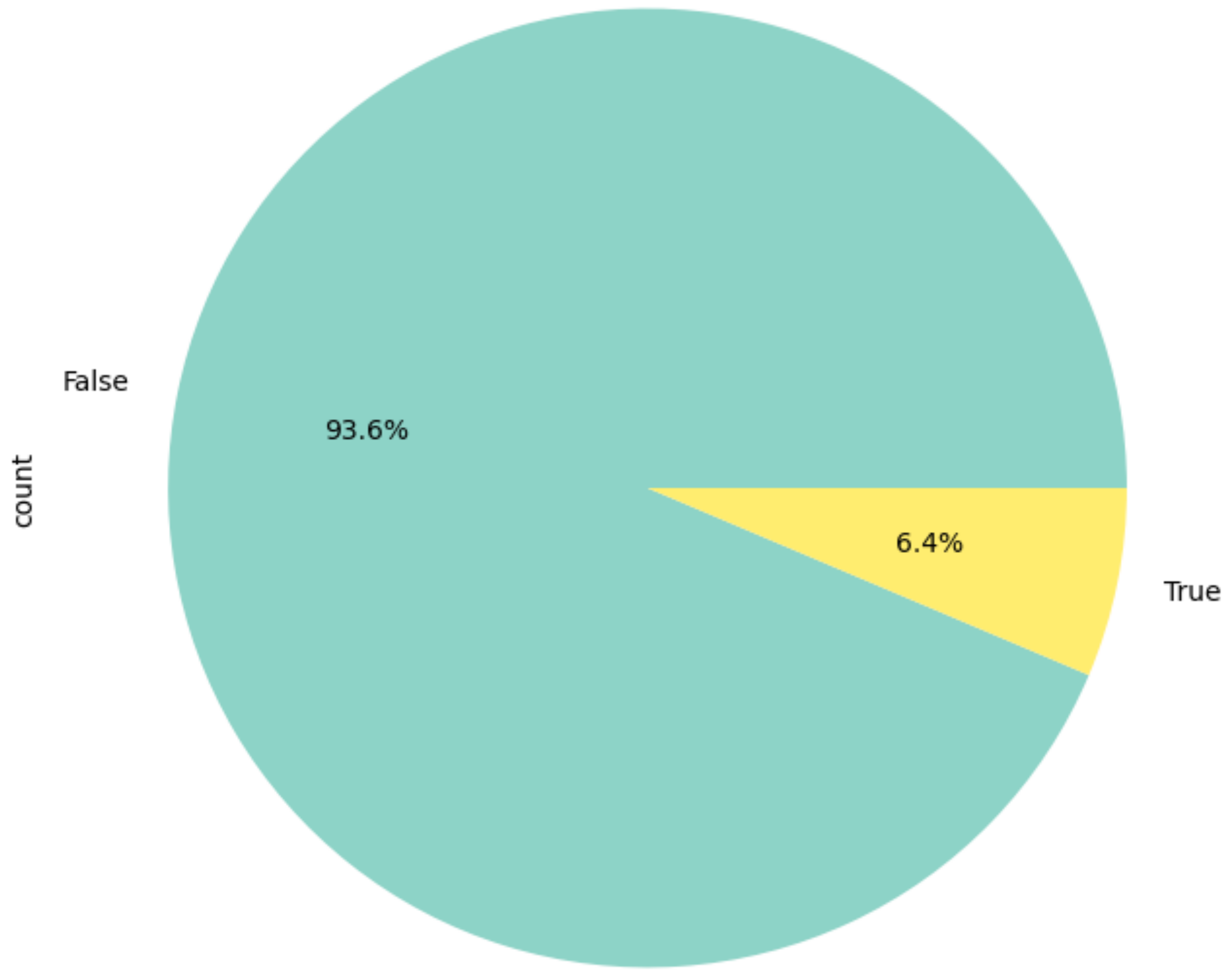
```
In [8]: df['Total'].plot(kind='box', vert=False, figsize=(10, 5))
Out[8]: <Axes: >
```





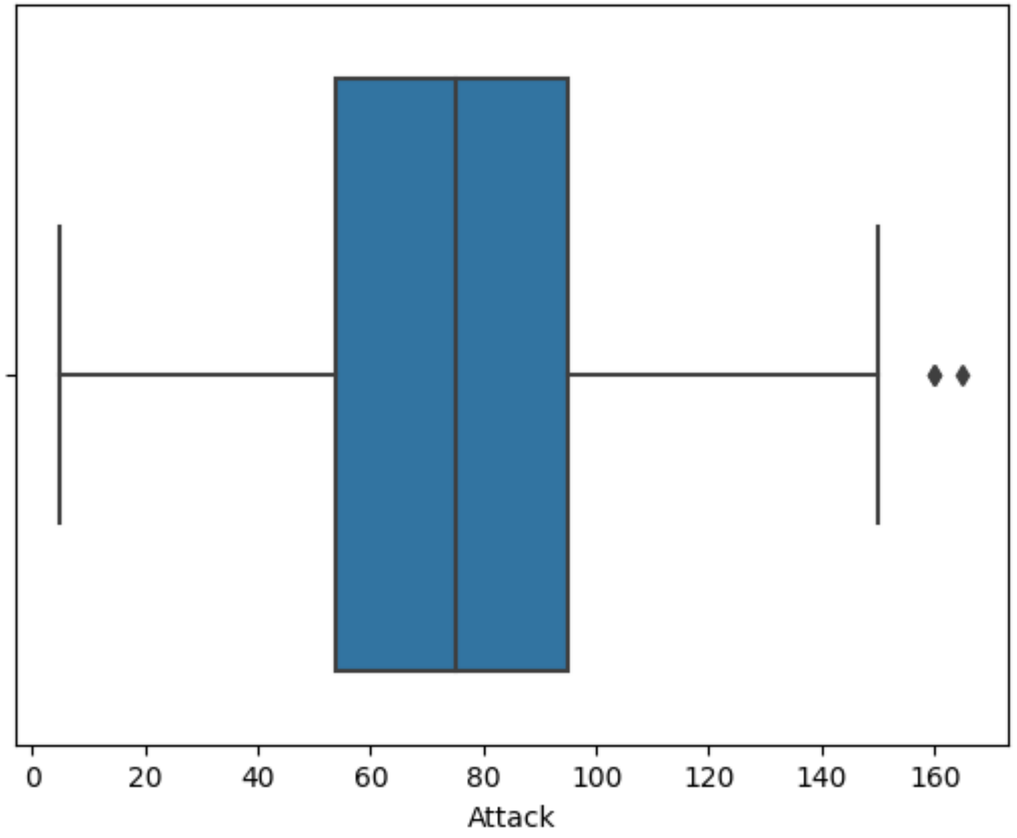
```
In [9]: df['Legendary'].value_counts().plot(kind='pie', autopct='%1.1f%%', cmap='Set3', figsize=(10, 8))

Out[9]: <Axes: ylabel='count'>
```

ıeir "Attack" feature). A boxplot is a



In [10]:
```
sns.boxplot(data=df, x='Attack')
```

Out[10]:
```
<Axes: xlabel='Attack'>
```



In [11]:
```
# Try your code here
df.loc[df['Attack'] > 150]
```

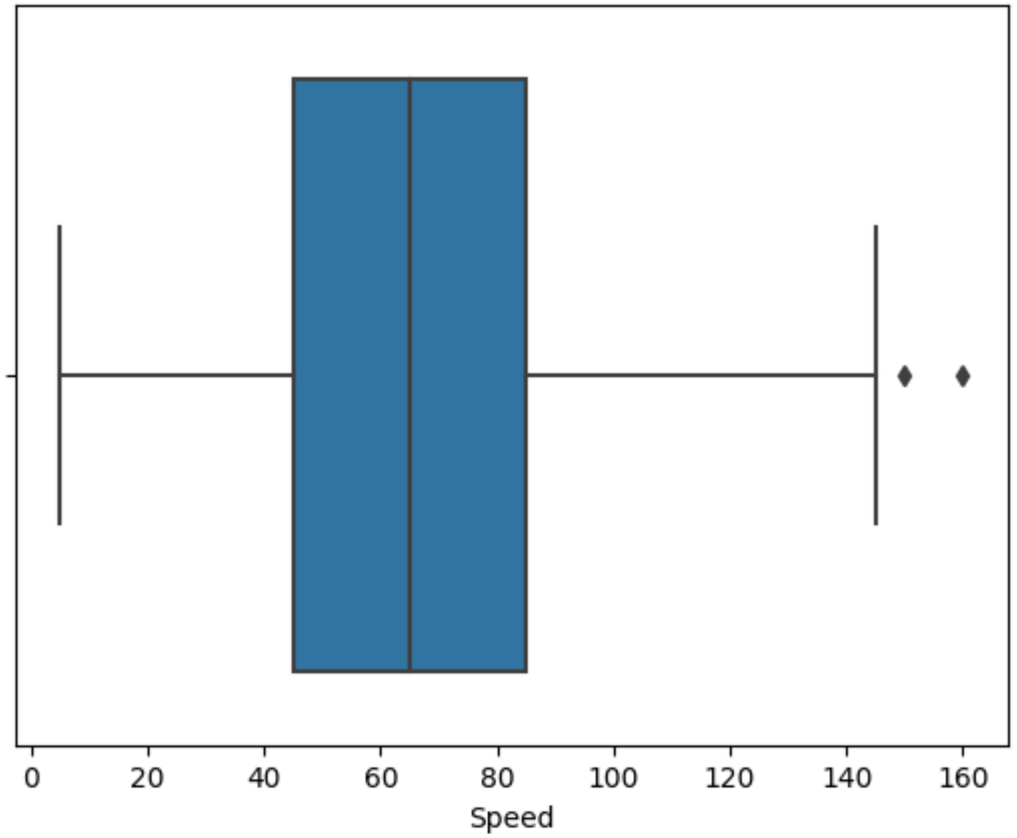
Out[11]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
288	289	Slaking	Normal	NaN	670	150	160	100	95	65	100	3	False
408	409	Rampardos	Rock	NaN	495	97	165	60	65	50	58	4	False
485	486	Regigigas	Normal	NaN	670	110	160	110	80	110	100	4	True

2. Select all pokemons with a Speed of `10` or less

```
In [12]: sns.boxplot(data=df, x='Speed')
```

```
Out[12]: <Axes: xlabel='Speed'>
```



```
In [14]: slow_pokemons_df = df.loc[df['Speed'] <=10]
```

3. How many Pokemons have a `Sp. Def` value of 25 or less?

In [15]:
```
# Try your code here
df.loc[df['Sp. Def'] <= 25]
```

Out[15]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
9	10	Caterpie	Bug	NaN	195	45	30	35	20	20	45	1	False
10	11	Metapod	Bug	NaN	205	50	20	55	25	25	30	1	False
12	13	Weedle	Bug	Poison	195	40	35	30	20	20	50	1	False
13	14	Kakuna	Bug	Poison	205	45	25	50	25	25	35	1	False
38	39	Jigglypuff	Normal	Fairy	270	115	45	20	45	25	20	1	False
89	90	Shellder	Water	NaN	305	30	65	100	45	25	40	1	False
97	98	Krabby	Water	NaN	325	30	105	90	25	25	50	1	False
115	116	Horsea	Water	NaN	295	30	40	70	70	25	60	1	False
128	129	Magikarp	Water	NaN	200	20	10	55	15	20	80	1	False
173	174	Igglybuff	Normal	Fairy	210	90	30	15	40	20	15	2	False
193	194	Wooper	Water	Ground	210	55	45	45	25	25	15	2	False
265	266	Silcoon	Bug	NaN	205	50	35	55	25	25	15	3	False
267	268	Cascoon	Bug	NaN	205	50	35	55	25	25	15	3	False
292	293	Whismur	Normal	NaN	240	64	51	23	51	23	28	3	False
317	318	Carvanha	Water	Dark	305	45	90	20	65	20	65	3	False
523	524	Roggenrola	Rock	NaN	280	55	75	85	25	25	15	5	False
663	664	Scatterbug	Bug	NaN	200	38	35	40	27	25	35	6	False

4. Select all the Legendary pokemons

In [16]:
```
# Try your code here
legendary_df = df[df['Legendary']]
legendary_df
```

Out[16]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
143	144	Articuno	Ice	Flying	580	90	85	100	95	125	85	1	True
144	145	Zapdos	Electric	Flying	580	90	90	85	125	90	100	1	True
145	146	Moltres	Fire	Flying	580	90	100	90	125	85	90	1	True
149	150	Mewtwo	Psychic	Fighting	680	106	110	90	154	90	130	1	True
242	243	Raikou	Electric	NaN	580	90	85	75	115	100	115	2	True
243	244	Entei	Fire	NaN	580	115	115	85	90	75	100	2	True
244	245	Suicune	Water	NaN	580	100	75	115	90	115	85	2	True
248	249	Lugia	Psychic	Flying	680	106	90	130	90	154	110	2	True
249	250	Ho-oh	Fire	Flying	680	106	130	90	110	154	90	2	True
376	377	Regirock	Rock	NaN	580	80	100	200	50	100	50	3	True
377	378	Regice	Ice	NaN	580	80	50	100	100	200	50	3	True
378	379	Registeel	Steel	NaN	580	80	75	150	75	150	50	3	True
379	380	Latias	Dragon	Psychic	600	80	80	90	110	130	110	3	True
380	381	Latios	Dragon	Psychic	600	80	90	80	130	110	110	3	True
381	382	Kyogre	Water	NaN	670	100	100	90	150	140	90	3	True
382	383	Groudon	Ground	Fire	670	100	150	140	100	90	90	3	True
383	384	Rayquaza	Dragon	Flying	680	105	150	90	150	90	95	3	True
384	385	Jirachi	Steel	Psychic	600	100	100	100	100	100	100	3	True
385	386	DeoxysNormal Forme	Psychic	NaN	600	50	150	50	150	50	150	3	True
479	480	Uxie	Psychic	NaN	580	75	75	130	75	130	95	4	True
480	481	Mesprit	Psychic	NaN	580	80	105	105	105	105	80	4	True
481	482	Azelf	Psychic	NaN	580	75	125	70	125	70	115	4	True

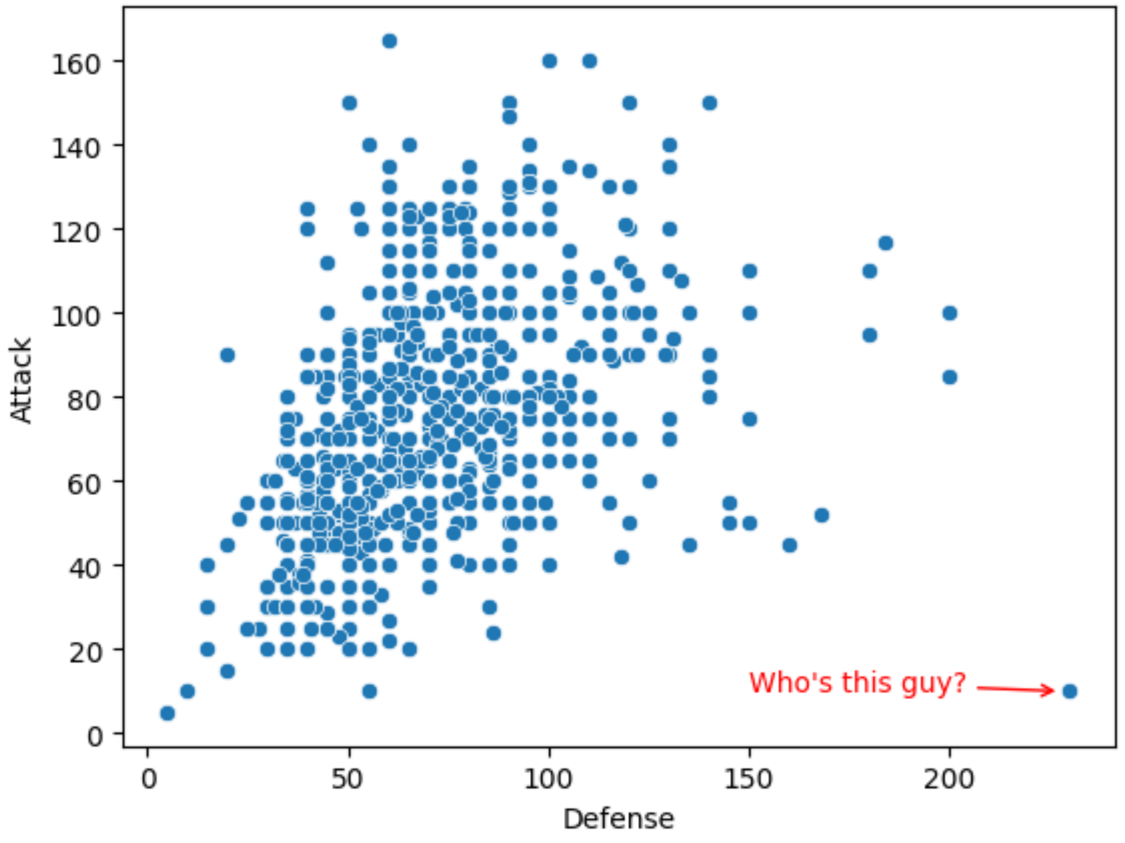
	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
482	483	Dialga	Steel	Dragon	680	100	120	120	150	100	90	4	True
483	484	Palkia	Water	Dragon	680	90	120	100	150	120	100	4	True
484	485	Heatran	Fire	Steel	600	91	90	106	130	106	77	4	True
485	486	Regigigas	Normal	NaN	670	110	160	110	80	110	100	4	True
486	487	GiratinaAltered Forme	Ghost	Dragon	680	150	100	120	100	120	90	4	True
490	491	Darkrai	Dark	NaN	600	70	90	90	135	90	125	4	True
491	492	ShayminLand Forme	Grass	Flying	600	100	100	100	100	100	100	4	True
492	493	Arceus	Normal	NaN	720	120	120	120	120	120	120	4	True
493	494	Victini	Psychic	Fire	600	100	100	100	100	100	100	5	True
637	638	Cobalion	Steel	Fighting	580	91	90	129	90	72	108	5	True
638	639	Terrakion	Rock	Fighting	580	91	129	90	72	90	108	5	True
639	640	Virizion	Grass	Fighting	580	91	90	72	90	129	108	5	True
640	641	TornadusIncarnate Forme	Flying	NaN	580	79	115	70	125	80	111	5	True
641	642	ThundurusIncarnate Forme	Electric	Flying	580	79	115	70	125	80	111	5	True
642	643	Reshiram	Dragon	Fire	680	100	120	100	150	120	90	5	True
643	644	Zekrom	Dragon	Electric	680	100	150	120	120	100	90	5	True
644	645	LandorusIncarnate Forme	Ground	Flying	600	89	125	90	115	80	101	5	True
645	646	Kyurem	Dragon	Ice	660	125	130	90	130	90	95	5	True
715	716	Xerneas	Fairy	NaN	680	126	131	95	131	98	99	6	True
716	717	Yveltal	Dark	Flying	680	126	131	95	131	98	99	6	True
717	718	Zygarde50% Forme	Dragon	Ground	600	108	100	121	81	95	95	6	True
718	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	True
719	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	True

5. Find the outlier

Find the pokemon that is clearly an outlier in terms of Attack / Defense:

```
In [18]: ax = sns.scatterplot(data=df, x="Defense", y="Attack")
         ax.annotate(
             "Who's this guy?", xy=(228, 10), xytext=(150, 10), color='red',
             arrowprops=dict(arrowstyle="->", color='red')
         )
```

```
Out[18]: Text(150, 10, "Who's this guy?")
```



```
In [26]: attack_cond = df['Attack'] < 20
In [27]: defence_cond = df['Defense'] > 200
```

```
In [29]: # Try your code here
         df.loc[attack_cond & defence_cond]
```

Out[29]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
212	213	Shuckle	Bug	Rock	505	20	10	230	10	230	5	2	False

Advanced selection

Now let's use boolean operators to create more advanced expressions

6. How many Fire-Flying Pokemons are there?

```
In [38]: # Try your code here
         df_query = df.query("`Type 1`=='Fire' and `Type 2`=='Flying'")
         df_query
```

Out[38]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
5	6	Charizard	Fire	Flying	534	78	84	78	109	85	100	1	False
145	146	Moltres	Fire	Flying	580	90	100	90	125	85	90	1	True
249	250	Ho-oh	Fire	Flying	680	106	130	90	110	154	90	2	True
661	662	Fletchinder	Fire	Flying	382	62	73	55	56	52	84	6	False
662	663	Talonflame	Fire	Flying	499	78	81	71	74	69	126	6	False

In [40]:
```
df.loc[((df['Type 1']=='Fire') & (df['Type 2']=='Flying'))]
```

Out[40]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
5	6	Charizard	Fire	Flying	534	78	84	78	109	85	100	1	False
145	146	Moltres	Fire	Flying	580	90	100	90	125	85	90	1	True
249	250	Ho-oh	Fire	Flying	680	106	130	90	110	154	90	2	True
661	662	Fletchinder	Fire	Flying	382	62	73	55	56	52	84	6	False
662	663	Talonflame	Fire	Flying	499	78	81	71	74	69	126	6	False

7. How many 'Poison' pokemons are across both types?

In [45]:
```
# Try your code here
df.query("`Type 1`=='Poison' or `Type 2`=='Poison'").shape
```

Out[45]:
```
(59, 13)
```

8. What pokemon of `Type 1` Ice has the strongest defense?

In [51]:
```
# Try your code here
ice_type1_pokemons = df.loc[df['Type 1']=='Ice']
```

In [55]:
```
df.loc[df['Defense'] == ice_type1_pokemons['Defense'].max()]
```

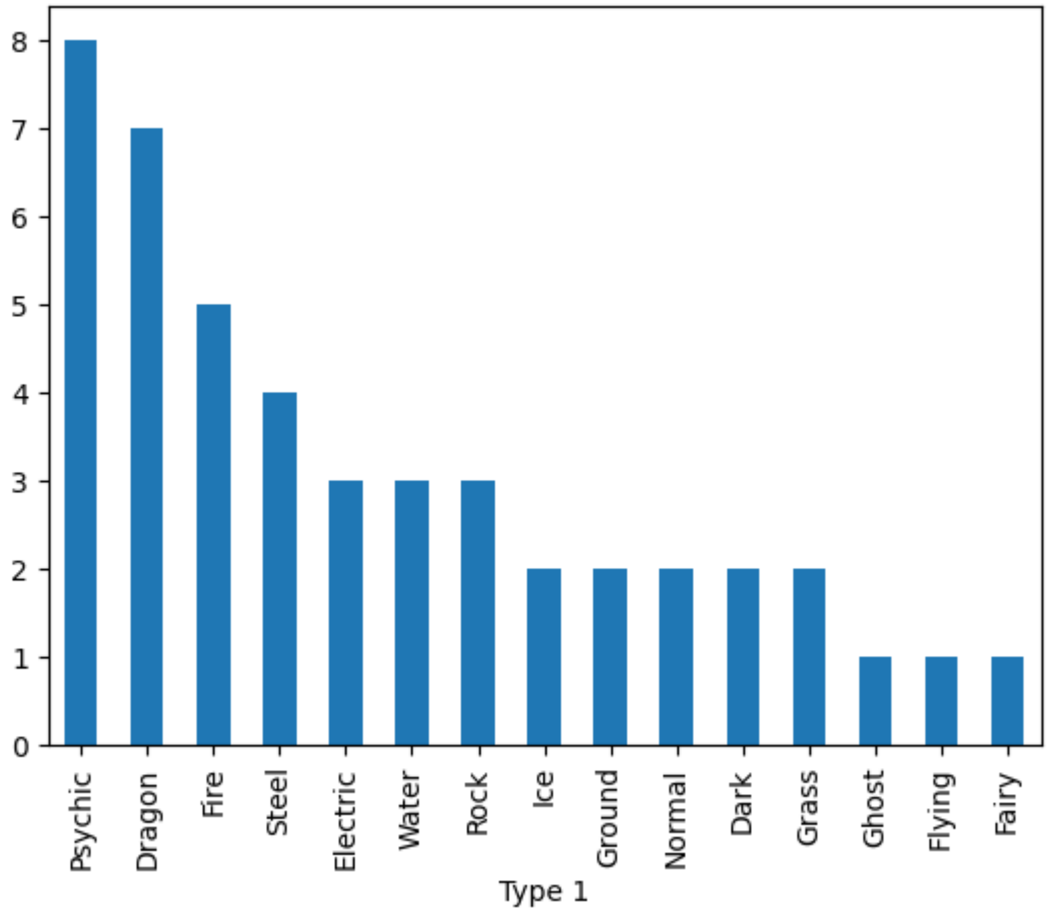
Out[55]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
712	713	Avalugg	Ice	NaN	514	95	117	184	44	46	28	6	False

9. What's the most common type of Legendary Pokemons?

In [66]:
```
# Try your code here
legendary_df = df.loc[df['Legendary']]
legendary_df['Type 1'].value_counts().plot(kind = 'bar')
```

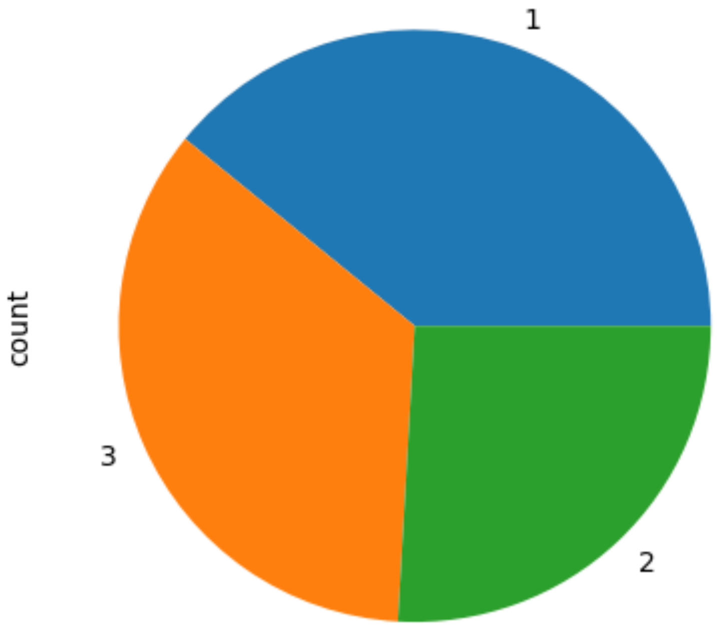
Out[66]:
```
<Axes: xlabel='Type 1'>
```



10. What's the most powerful pokemon from the first 3 generations, of type water?

In [69]:
```
# Try your code here
df_3_gens = df.loc[df['Generation'].isin([1,2,3])]
df_3_gens['Generation'].value_counts().plot(kind='pie')
```

Out[69]:
```
<Axes: ylabel='count'>
```



In [73]:
```
df_3_gens_water = df_3_gens.query("`Type 1`=='Water' or `Type 2`=='Water'")
df_3_gens_water.loc[df_3_gens_water['Total']==df_3_gens_water['Total'].max()]
```

Out[73]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
381	382	Kyogre	Water	NaN	670	100	100	90	150	140	90	3	True

In [74]:
```
df_3_gens_water.sort_values(by='Total', ascending=False)
```

Out[74]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
381	382	Kyogre	Water	NaN	670	100	100	90	150	140	90	3	True
244	245	Suicune	Water	NaN	580	100	75	115	90	115	85	2	True
349	350	Milotic	Water	NaN	540	95	60	79	100	125	81	3	False
229	230	Kingdra	Water	Dragon	540	75	95	95	95	95	85	2	False
129	130	Gyarados	Water	Flying	540	95	125	79	60	100	81	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
182	183	Marill	Water	Fairy	250	70	20	50	20	50	40	2	False
269	270	Lotad	Water	Grass	220	40	30	30	40	50	30	3	False
193	194	Wooper	Water	Ground	210	55	45	45	25	25	15	2	False
128	129	Magikarp	Water	NaN	200	20	10	55	15	20	80	1	False
348	349	Feebas	Water	NaN	200	20	15	20	10	55	80	3	False

78 rows × 13 columns

11. What's the most powerful Dragon from the last two generations?

In [75]:
```
# Try your code here
df['Generation'].value_counts()
```

Out[75]:
```
Generation
5    156
1    151
3    135
4    107
2    100
6     72
Name: count, dtype: int64
```

In [87]:
```
df.loc[((df['Generation'].isin([5,6])
        & ((df['Type 1']=='Dragon') | (df['Type 2']=='Dragon'))
       ))].sort_values(by='Total', ascending=False)
```

Out[87]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
643	644	Zekrom	Dragon	Electric	680	100	150	120	120	100	90	5	True
642	643	Reshiram	Dragon	Fire	680	100	120	100	150	120	90	5	True
645	646	Kyurem	Dragon	Ice	660	125	130	90	130	90	95	5	True
634	635	Hydreigon	Dark	Dragon	600	92	105	90	125	90	98	5	False
705	706	Goodra	Dragon	NaN	600	90	100	70	110	150	80	6	False
717	718	Zygarde50% Forme	Dragon	Ground	600	108	100	121	81	95	95	6	True
611	612	Haxorus	Dragon	NaN	540	76	147	90	60	70	97	5	False
714	715	Noivern	Flying	Dragon	535	85	70	80	97	80	123	6	False
696	697	Tyrantrum	Rock	Dragon	521	82	121	119	69	59	71	6	False
690	691	Dragalge	Poison	Dragon	494	65	75	90	97	123	44	6	False
620	621	Druddigon	Dragon	NaN	485	77	120	90	60	90	48	5	False
704	705	Sliggoo	Dragon	NaN	452	68	75	53	83	113	60	6	False
633	634	Zweilous	Dark	Dragon	420	72	85	70	65	70	58	5	False
610	611	Fraxure	Dragon	NaN	410	66	117	70	40	50	67	5	False
695	696	Tyrunt	Rock	Dragon	362	58	89	77	45	45	48	6	False
609	610	Axew	Dragon	NaN	320	46	87	60	30	40	57	5	False
632	633	Deino	Dark	Dragon	300	52	65	50	45	50	38	5	False
703	704	Goomy	Dragon	NaN	300	45	50	35	55	75	40	6	False
713	714	Noibat	Flying	Dragon	245	40	30	35	45	40	55	6	False

12. Select most powerful Fire-type pokemons

In [99]:
```
# Try your code here
powerful_fire_df = df.query("`Type 1`=='Fire' and `Attack`>100")
powerful_fire_df
```

Out[99]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
58	59	Arcanine	Fire	NaN	555	90	110	80	100	80	95	1	False
135	136	Flareon	Fire	NaN	525	65	130	60	95	110	65	1	False
243	244	Entei	Fire	NaN	580	115	115	85	90	75	100	2	True
249	250	Ho-oh	Fire	Flying	680	106	130	90	110	154	90	2	True
256	257	Blaziken	Fire	Fighting	530	80	120	70	110	70	80	3	False
391	392	Infernape	Fire	Fighting	534	76	104	71	104	71	108	4	False
499	500	Emboar	Fire	Fighting	528	110	123	65	100	65	65	5	False
554	555	DarmanitanStandard Mode	Fire	Psychic	480	105	140	55	30	55	95	5	False
720	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True

13. Select all Water-type, Flying-type pokemons

```
In [104]: # Try your code here
          water_flying_df = df.query("`Type 1`=='Water' and `Type 2`=='Flying'")
          water_flying_df
```

Out[104]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
129	130	Gyarados	Water	Flying	540	95	125	79	60	100	81	1	False
225	226	Mantine	Water	Flying	465	65	40	70	80	140	70	2	False
277	278	Wingull	Water	Flying	270	40	30	30	55	30	85	3	False
278	279	Pelipper	Water	Flying	430	60	50	100	85	70	65	3	False
457	458	Mantyke	Water	Flying	345	45	20	50	60	120	50	4	False
579	580	Ducklett	Water	Flying	305	62	44	50	44	50	55	5	False
580	581	Swanna	Water	Flying	473	75	87	63	87	63	98	5	False

14. Select specific columns of Legendary pokemons of type Fire

```
In [117]: # Try your code here
          legendary_fire_df = df.loc[( (df['Type 1']=='Fire') & (df['Legendary']) )][['Name','Attack','Generation']]
          legendary_fire_df
```

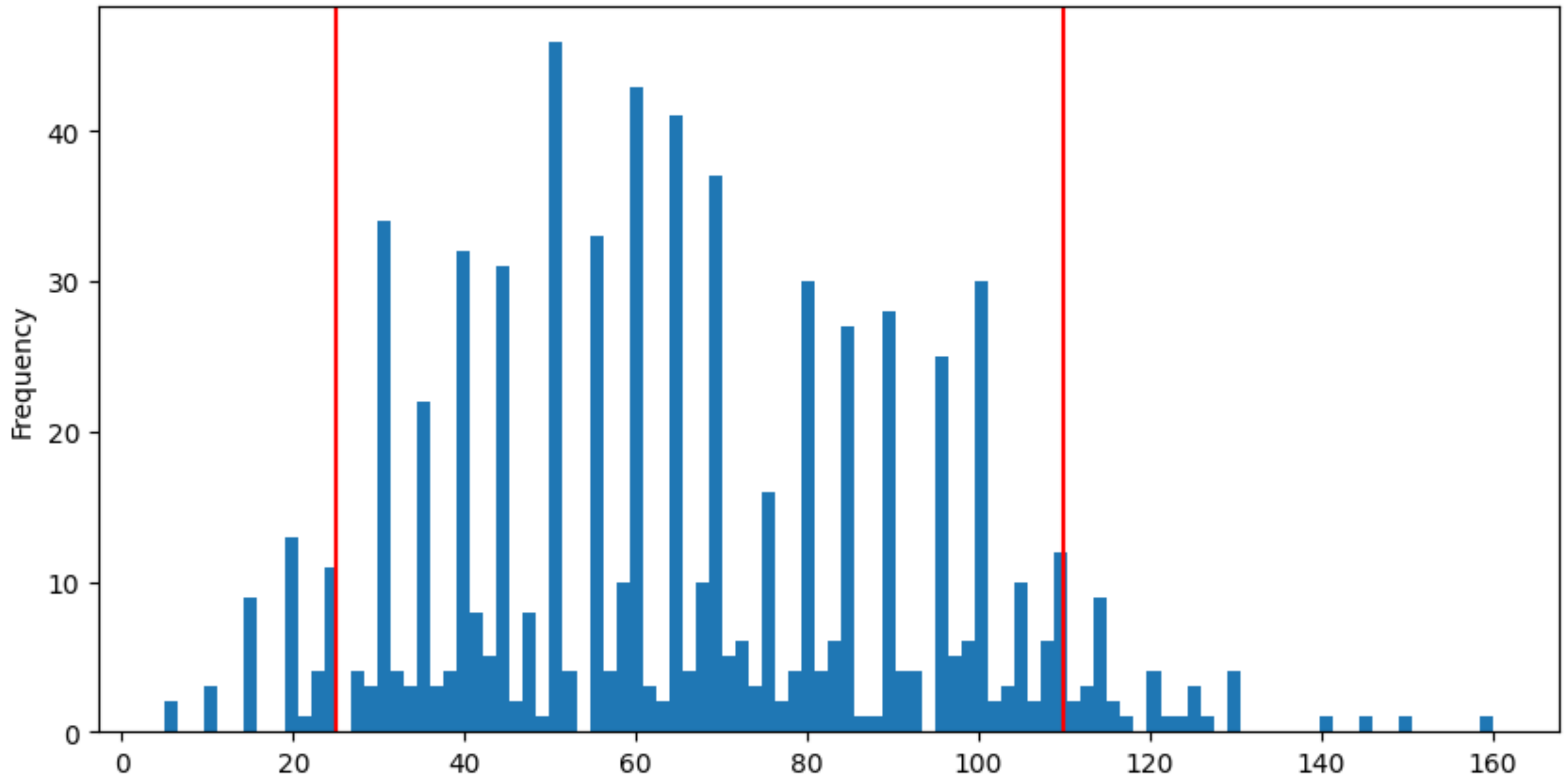
Out[117]:

	Name	Attack	Generation
145	Moltres	100	1
243	Entei	115	2
249	Ho-oh	130	2
484	Heatran	90	4
720	Volcanion	110	6

15. Select Slow and Fast pokemons

This is the distribution of speed of the pokemons. The red lines indicate those bottom 5% and top 5% pokemons by speed:

```
In [119]: ax = df['Speed'].plot(kind='hist', figsize=(10, 5), bins=100)
          ax.axvline(df['Speed'].quantile(.05), color='red')
          ax.axvline(df['Speed'].quantile(.95), color='red')

Out[119]: <matplotlib.lines.Line2D at 0x7f492e54e910>
```



```
In [120]: df['Speed'].describe()

Out[120]: count    721.000000
          mean      65.714286
          std       27.277920
          min        5.000000
          25%       45.000000
          50%       65.000000
          75%       85.000000
          max      160.000000
          Name: Speed, dtype: float64
```

```
In [135]: bottom_5 = df['Speed'].quantile(.05)
          bottom_5

Out[135]: 25.0
```

```
In [136]: top_5 = df['Speed'].quantile(.95)
          top_5

Out[136]: 110.0
```

In [137]:
```
# Try your code here
slow_fast_df = df.loc[(df['Speed'] <bottom_5) | (df['Speed'] > top_5)]
slow_fast_df
```

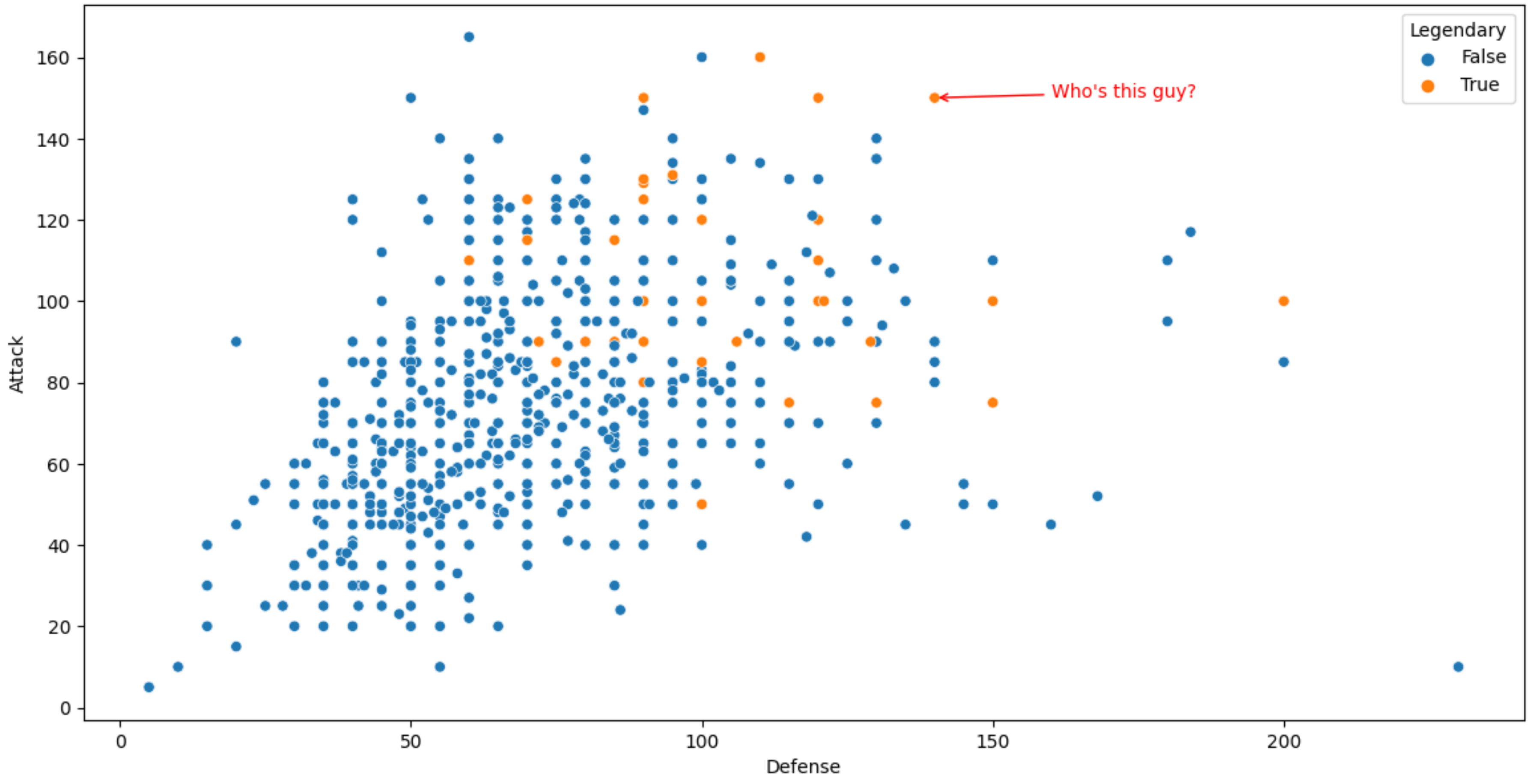
Out[137]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
38	39	Jigglypuff	Normal	Fairy	270	115	45	20	45	25	20	1	False
50	51	Dugtrio	Ground	NaN	405	35	80	50	50	70	120	1	False
52	53	Persian	Normal	NaN	440	65	70	60	65	65	115	1	False
64	65	Alakazam	Psychic	NaN	500	55	50	45	135	95	120	1	False
73	74	Geodude	Rock	Ground	300	40	80	100	30	30	20	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
657	658	Greninja	Water	Dark	530	72	95	67	103	71	122	6	False
662	663	Talonflame	Fire	Flying	499	78	81	71	74	69	126	6	False
681	682	Spritzee	Fairy	NaN	341	78	52	60	63	65	23	6	False
700	701	Hawlucha	Fighting	Flying	500	78	92	75	74	63	118	6	False
714	715	Noivern	Flying	Dragon	535	85	70	80	97	80	123	6	False

68 rows × 13 columns

16. Find the Ultra Powerful Legendary Pokemon

```
In [139]: fig, ax = plt.subplots(figsize=(14, 7))
          sns.scatterplot(data=df, 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')
          )

Out[139]: Text(160, 150, "Who's this guy?")
```



In [140]:
```
# Try your code here
df.query("`Attack` > 140 and `Defense` > 100")
```

Out[140]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
382	383	Groudon	Ground	Fire	670	100	150	140	100	90	90	3	True
485	486	Regigigas	Normal	NaN	670	110	160	110	80	110	100	4	True
643	644	Zekrom	Dragon	Electric	680	100	150	120	120	100	90	5	True

The End!