In [4]:

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

In [5]:

```
#importing data using pandas
pokeman = pd.read_csv(r"C:\Users\SamDutse\Documents\DATA\pokemon_data.csv")
```

In [6]:

```
#checking the top two rwos of the data
pokeman.head(2)
```

Out[6]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False

In [7]:

```
#checking the last two rows of the dataset
pokeman.tail(2)
```

Out[7]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
798	720	HoopaHoopa Unbound	Psychic	Dark	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	80	110	120	130	90	70	6	True

In [9]:

```
#checking the number of rows and column of data
pokeman.shape
```

Out[9]:

```
(800, 12)
```

In [10]:

```
#getting sntire information about each colum of or data
pokeman.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 12 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   HP          800 non-null    int64
 5   Attack      800 non-null    int64
 6   Defense     800 non-null    int64
 7   Sp. Atk     800 non-null    int64
 8   Sp. Def     800 non-null    int64
 9   Speed       800 non-null    int64
 10  Generation  800 non-null    int64
 11  Legendary   800 non-null    bool
dtypes: bool(1), int64(8), object(3)
memory usage: 69.7+ KB
```

In [11]:

```
#from the above the Type 2 column has 414 non null columns out of 800
#confirming the nulls in the pokeman data
pokeman.isnull().sum()
```

Out[11]:

```
#               0
Name            0
Type 1          0
Type 2        386
HP              0
Attack          0
Defense         0
Sp. Atk         0
Sp. Def         0
Speed           0
Generation      0
Legendary       0
dtype: int64
```

In [12]:

```
pokeman.nunique()
```

Out[12]:

```
#             721
Name          800
Type 1         18
Type 2         18
HP             94
Attack        111
Defense       103
Sp. Atk       105
Sp. Def        92
Speed         108
Generation      6
Legendary       2
dtype: int64
```

In [13]:

```
pokeman.dtypes
```

Out[13]:

```
#              int64
Name          object
Type 1        object
Type 2        object
HP             int64
Attack         int64
Defense        int64
Sp. Atk        int64
Sp. Def        int64
Speed          int64
Generation     int64
Legendary       bool
dtype: object
```

In [14]:

```
'''we would have use pokeman["Type 2"].fillna(pokeman[["Type 2"]].mean())
to fill the missing value but it is not a numerical column'''
pokeman
```

Out[14]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	50	100	150	100	150	50	6	True

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
796	719	DiancieMega Diancie	Rock	Fairy	50	160	110	160	110	110	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	80	110	60	150	130	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	80	110	120	130	90	70	6	True

800 rows × 12 columns

In [15]:

```
pokeman.describe().transpose()
```

Out[15]:

	count	mean	std	min	25%	50%	75%	max
#	800.0	362.81375	208.343798	1.0	184.75	364.5	539.25	721.0
HP	800.0	69.25875	25.534669	1.0	50.00	65.0	80.00	255.0
Attack	800.0	79.00125	32.457366	5.0	55.00	75.0	100.00	190.0
Defense	800.0	73.84250	31.183501	5.0	50.00	70.0	90.00	230.0
Sp. Atk	800.0	72.82000	32.722294	10.0	49.75	65.0	95.00	194.0
Sp. Def	800.0	71.90250	27.828916	20.0	50.00	70.0	90.00	230.0
Speed	800.0	68.27750	29.060474	5.0	45.00	65.0	90.00	180.0
Generation	800.0	3.32375	1.661290	1.0	2.00	3.0	5.00	6.0

In [17]:

```
#checking pokeman with weakest attack
pokeman_Non_na=pokeman.dropna()
```

In [18]:

```
pokeman_Non_na
```

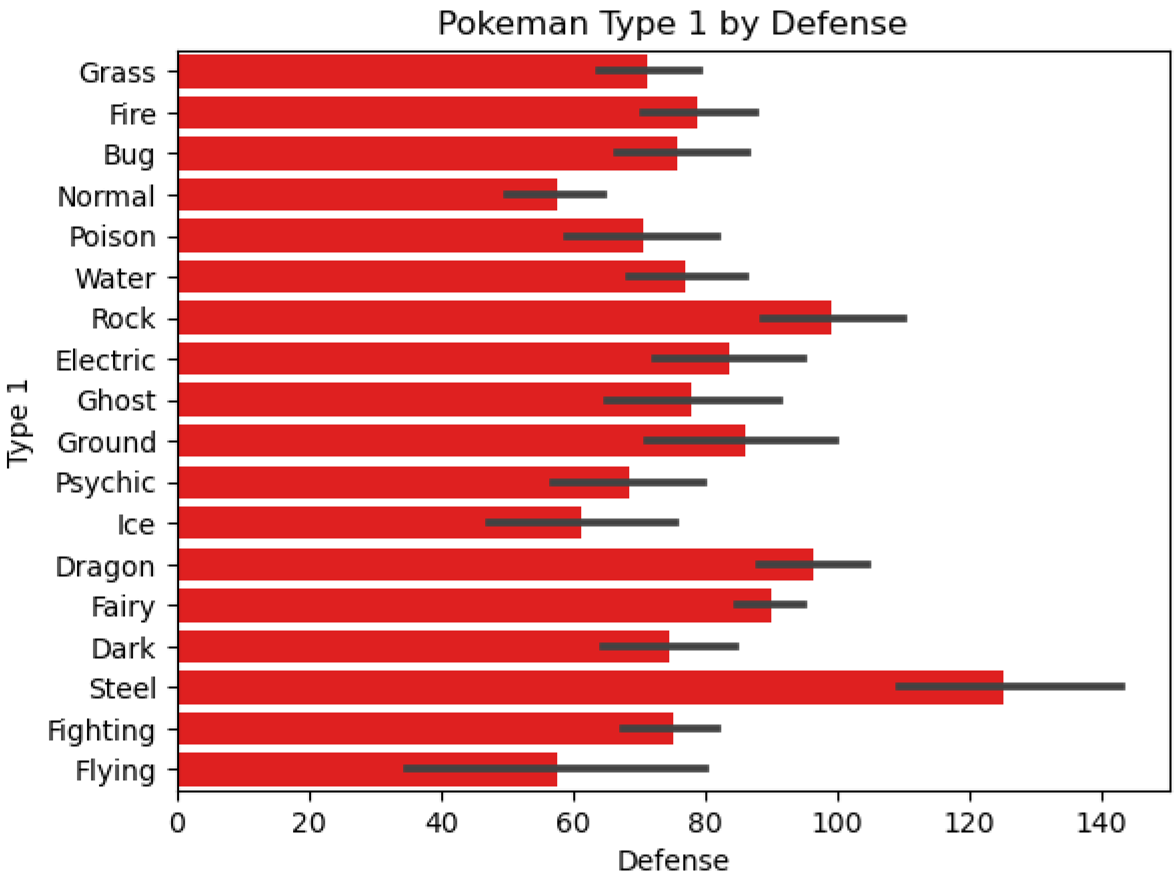
Out[18]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
6	6	Charizard	Fire	Flying	78	84	78	109	85	100	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	50	100	150	100	150	50	6	True
796	719	DiancieMega Diancie	Rock	Fairy	50	160	110	160	110	110	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	80	110	60	150	130	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	80	110	120	130	90	70	6	True

414 rows × 12 columns

In [19]:
```
sns.barplot(x = 'Defense', y = 'Type 1', data = pokeman_Non_na, color = 'red').set(title
```

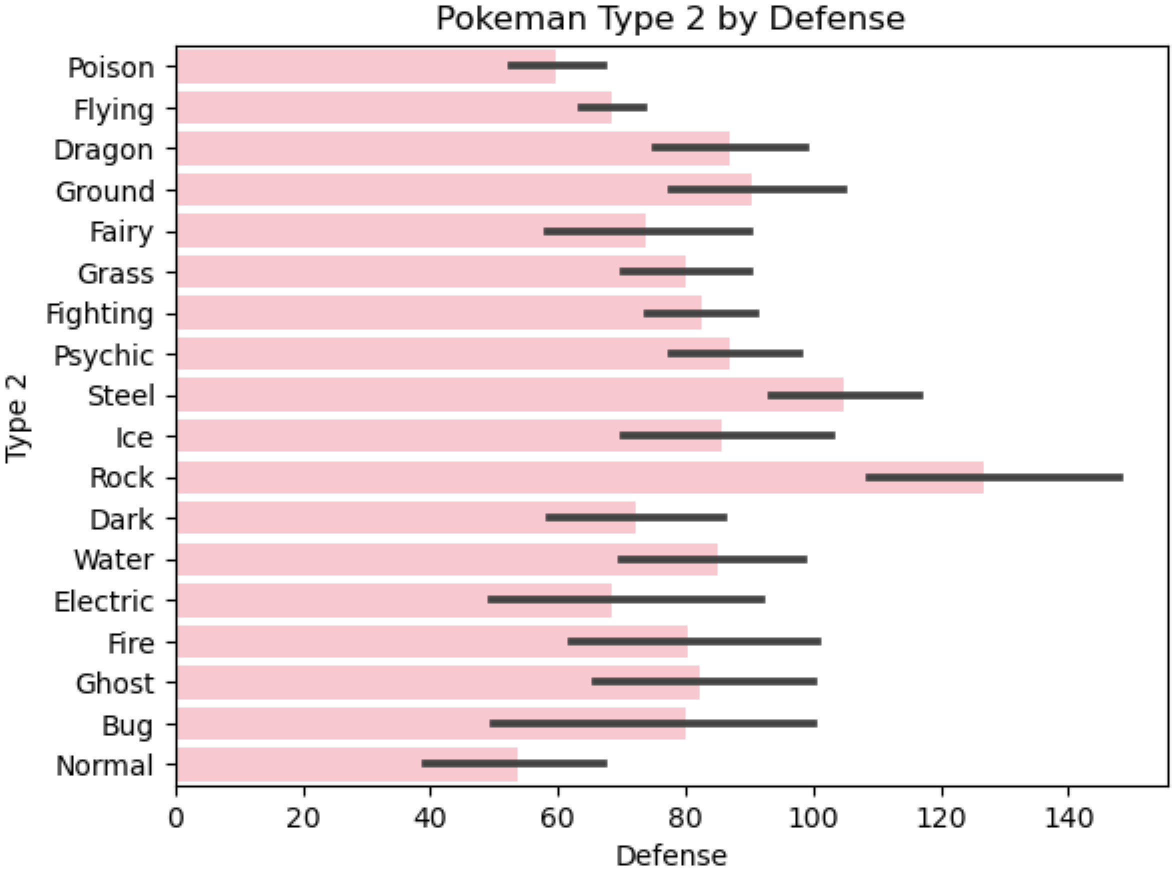
Out[19]:
```
[Text(0.5, 1.0, 'Pokeman Type 1 by Defense')]
```



The above shows that the Pokeman Type 1 with the highest Defense is Rock and the one with the weakest Defense are two (Flying and Normal)

In [21]:
```
sns.barplot(x = 'Defense', y = 'Type 2', data = pokeman_Non_na, color = 'pink').set(titl
```

Out[21]:
```
[Text(0.5, 1.0, 'Pokeman Type 2 by Defense')]
```



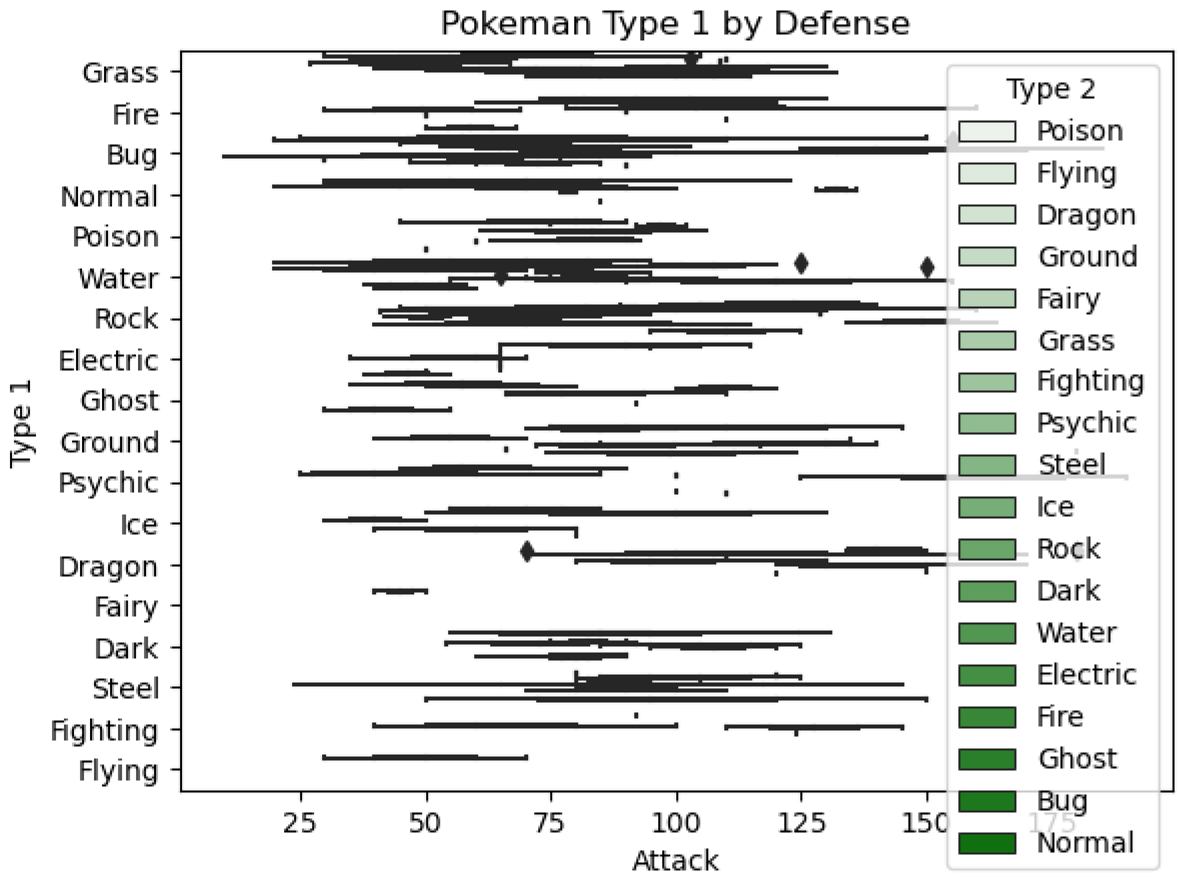
The above shows that the Pokeman Type 2 with the highest Defense is Rock and the one with the weakest Defense is Normal

In [22]:

```
sns.boxplot(x = 'Attack', y = 'Type 1', data = pokeman_Non_na, color = 'green', hue='Typ
```

Out[22]:

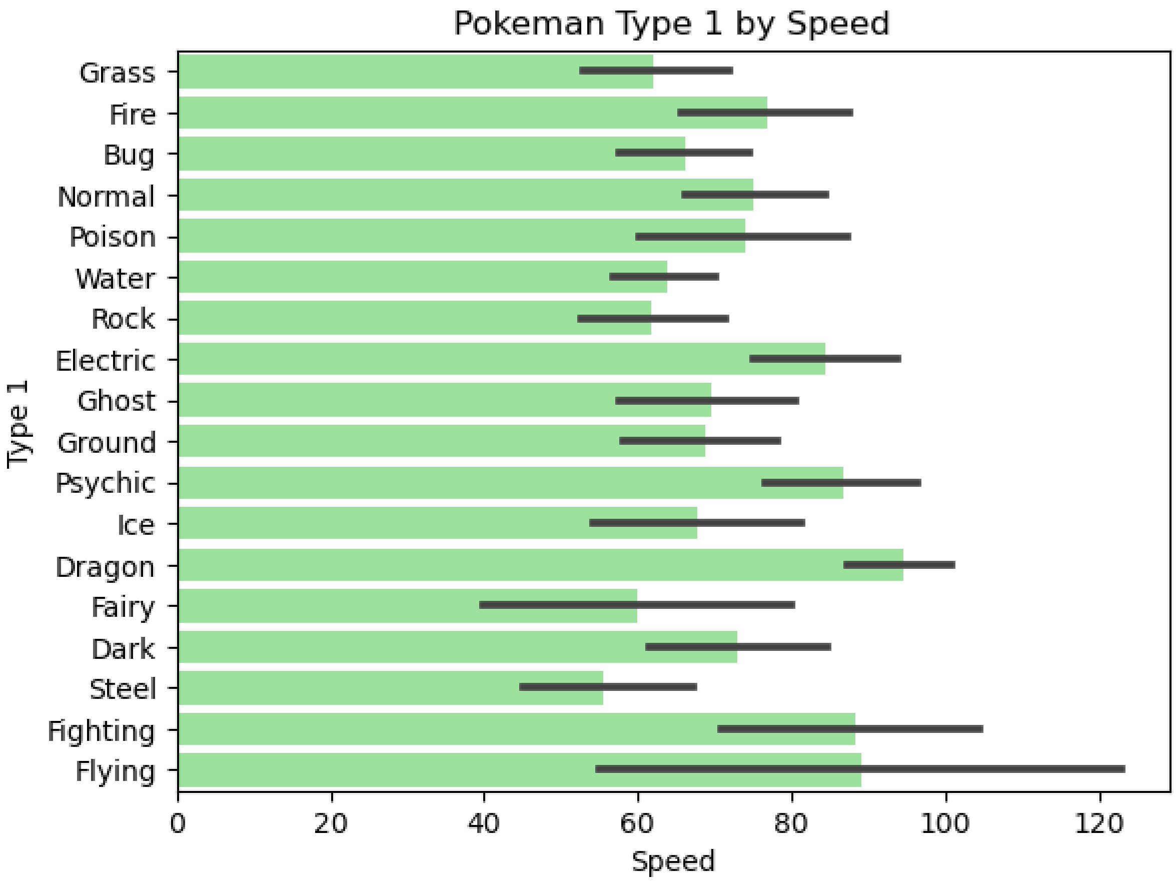
```
[Text(0.5, 1.0, 'Pokeman Type 1 by Defense')]
```



The above boxplot indicate that the Pokeman Type 1 and Type 2 have the same Attack of 5

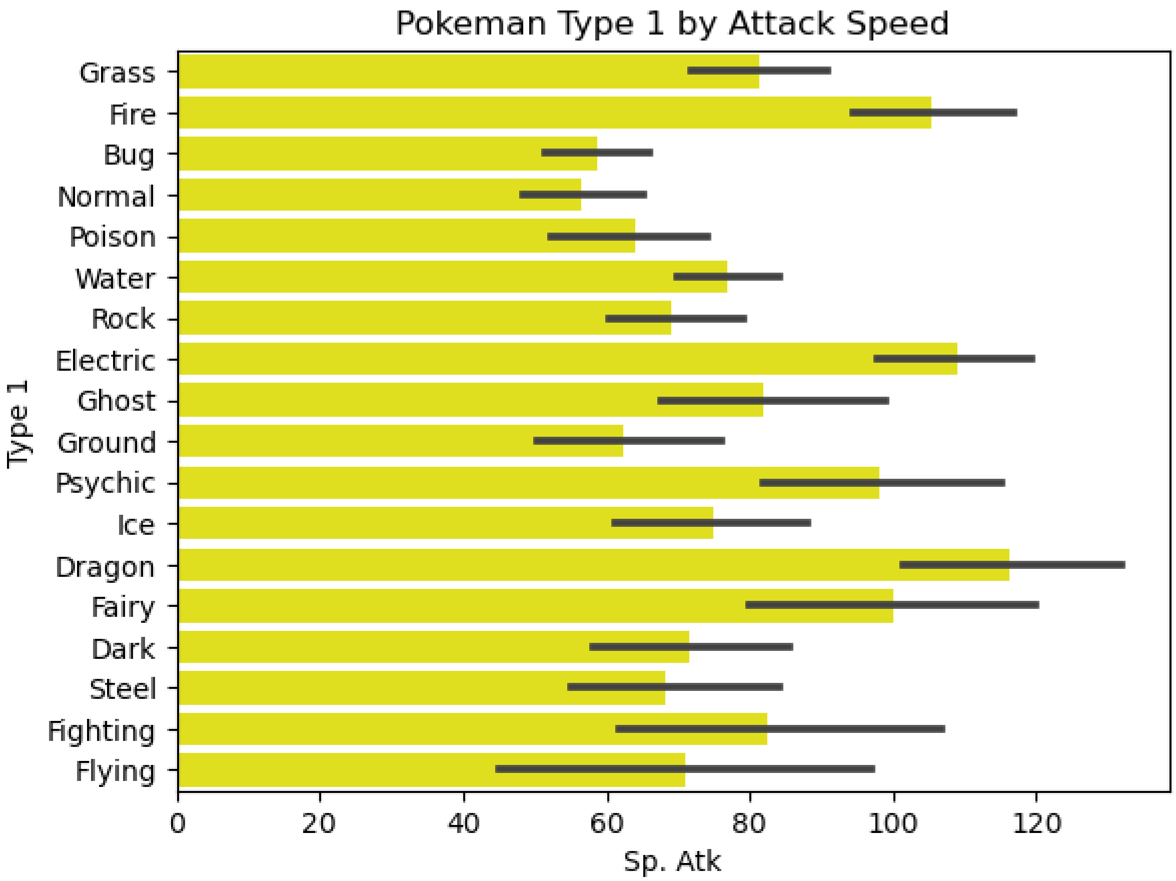
```
In [23]: sns.barplot(x = 'Speed', y = 'Type 1', data = pokeman_Non_na, color = 'lightgreen').set(
```

```
Out[23]: [Text(0.5, 1.0, 'Pokeman Type 1 by Speed')]
```



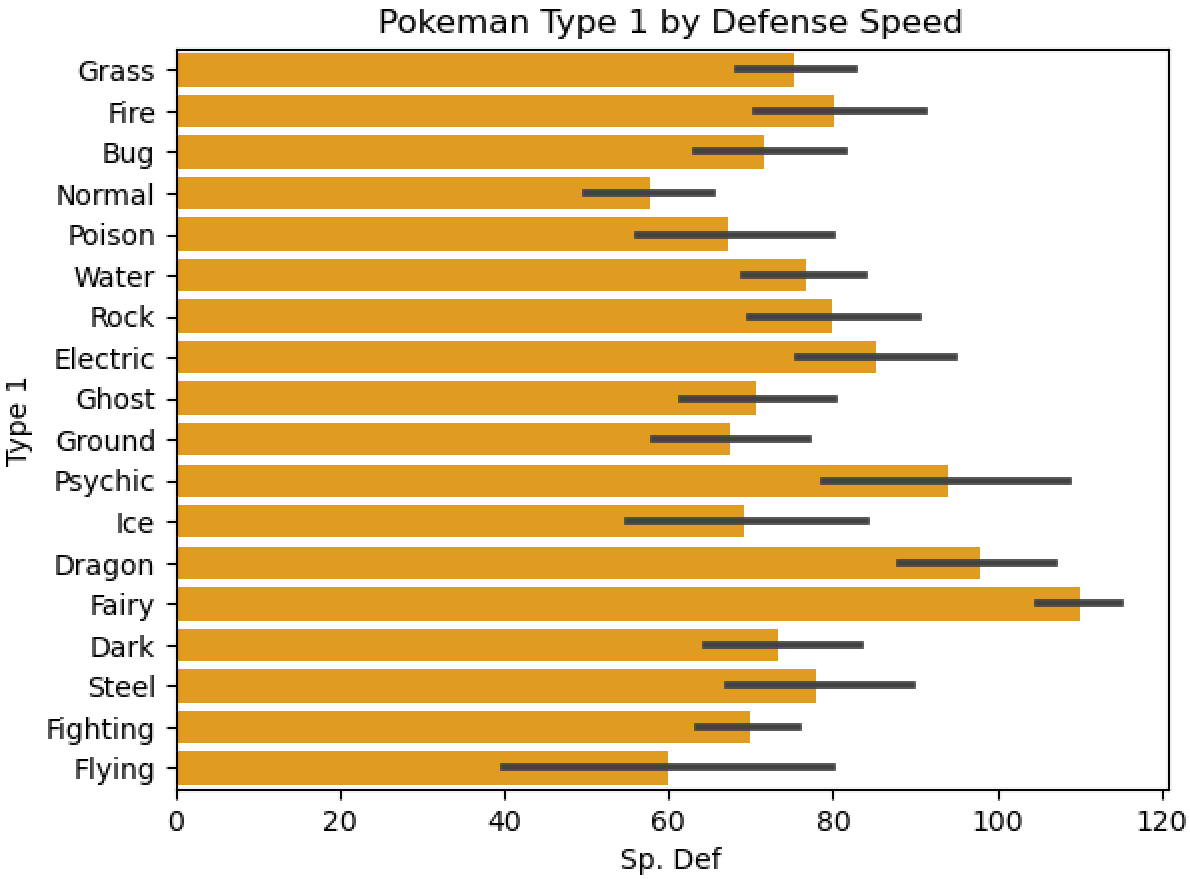
In [24]:
```
sns.barplot(x = 'Sp. Atk', y = 'Type 1', data = pokeman_Non_na, color = 'yellow').set(ti
```

Out[24]:
```
[Text(0.5, 1.0, 'Pokeman Type 1 by Attack Speed')]
```



In [25]:
```
sns.barplot(x = 'Sp. Def', y = 'Type 1', data = pokeman_Non_na, color = 'orange').set(ti
```

Out[25]:
```
[Text(0.5, 1.0, 'Pokeman Type 1 by Defense Speed')]
```



In terms of Speed, Attack speeed and Defense speed the best and weakest pokeman Type 1 respectively are;

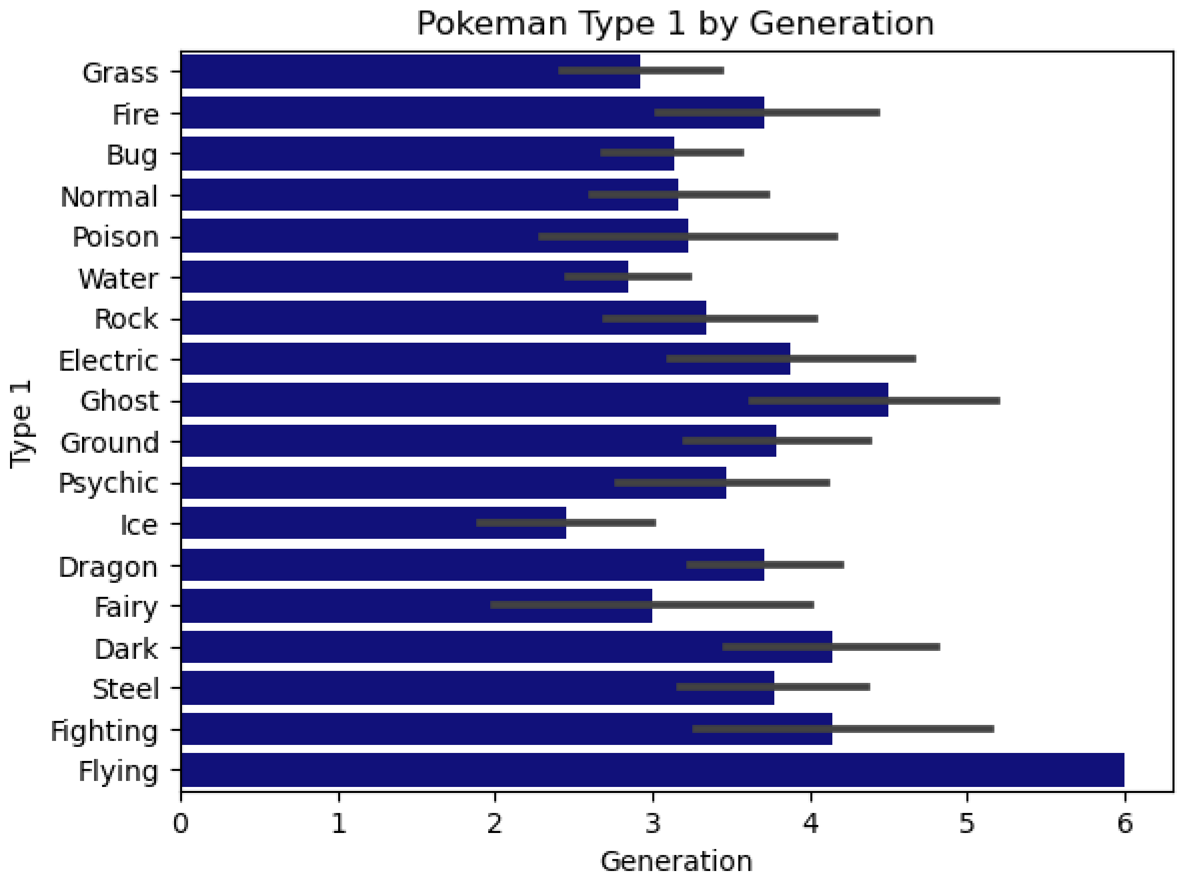
1. Speed = (Flying, steel)

2. Attack speed = (Dragon, Normal)

3. Defense speed = (Fairy, Poison)

```
In [26]: sns.barplot(x = 'Generation', y = 'Type 1', data = pokeman_Non_na, color = 'darkblue').s
```

```
Out[26]: [Text(0.5, 1.0, 'Pokeman Type 1 by Generation')]
```

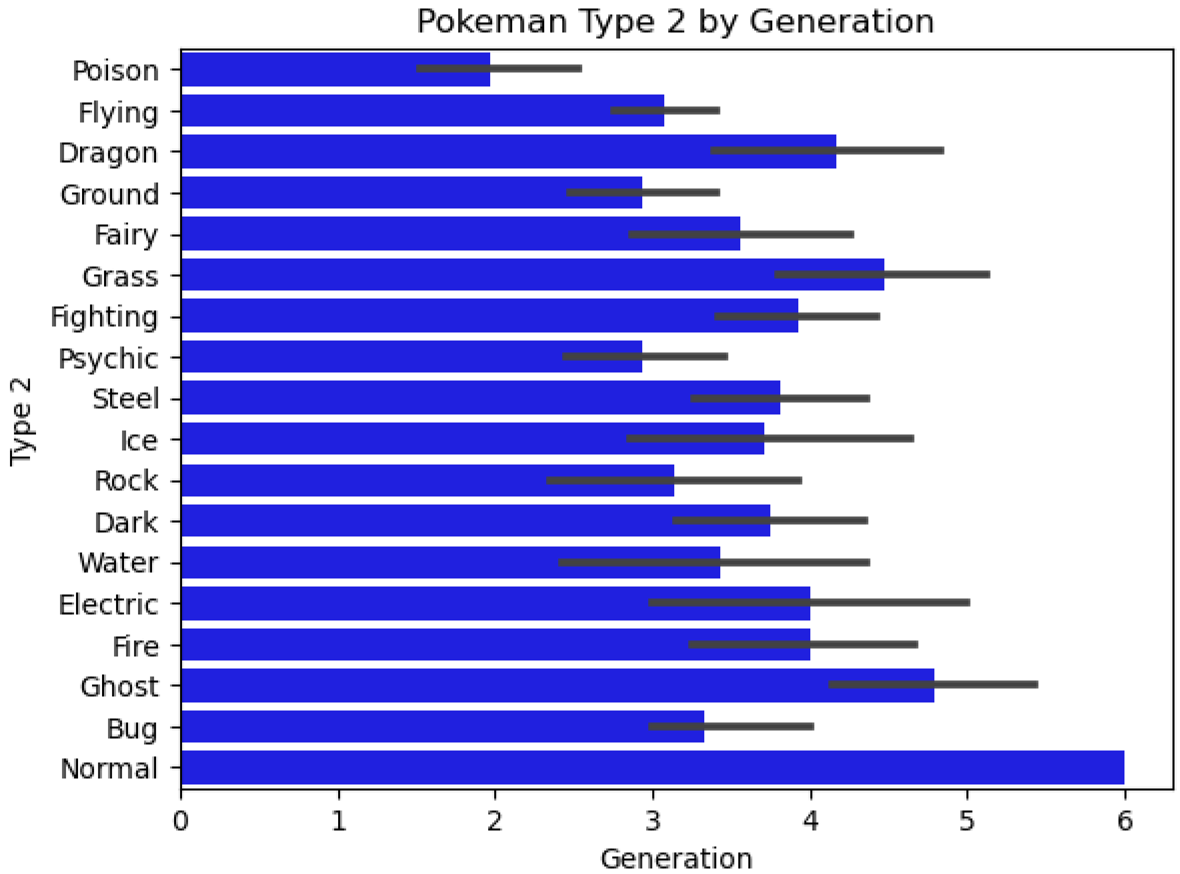


In [27]:

```
sns.barplot(x = 'Generation', y = 'Type 2', data = pokeman_Non_na, color = 'blue').set(t
```

Out[27]:

```
[Text(0.5, 1.0, 'Pokeman Type 2 by Generation')]
```



The two charts above shows the diffrent pokeman Type 1 and Type 2 by Generation ranging from 1-6

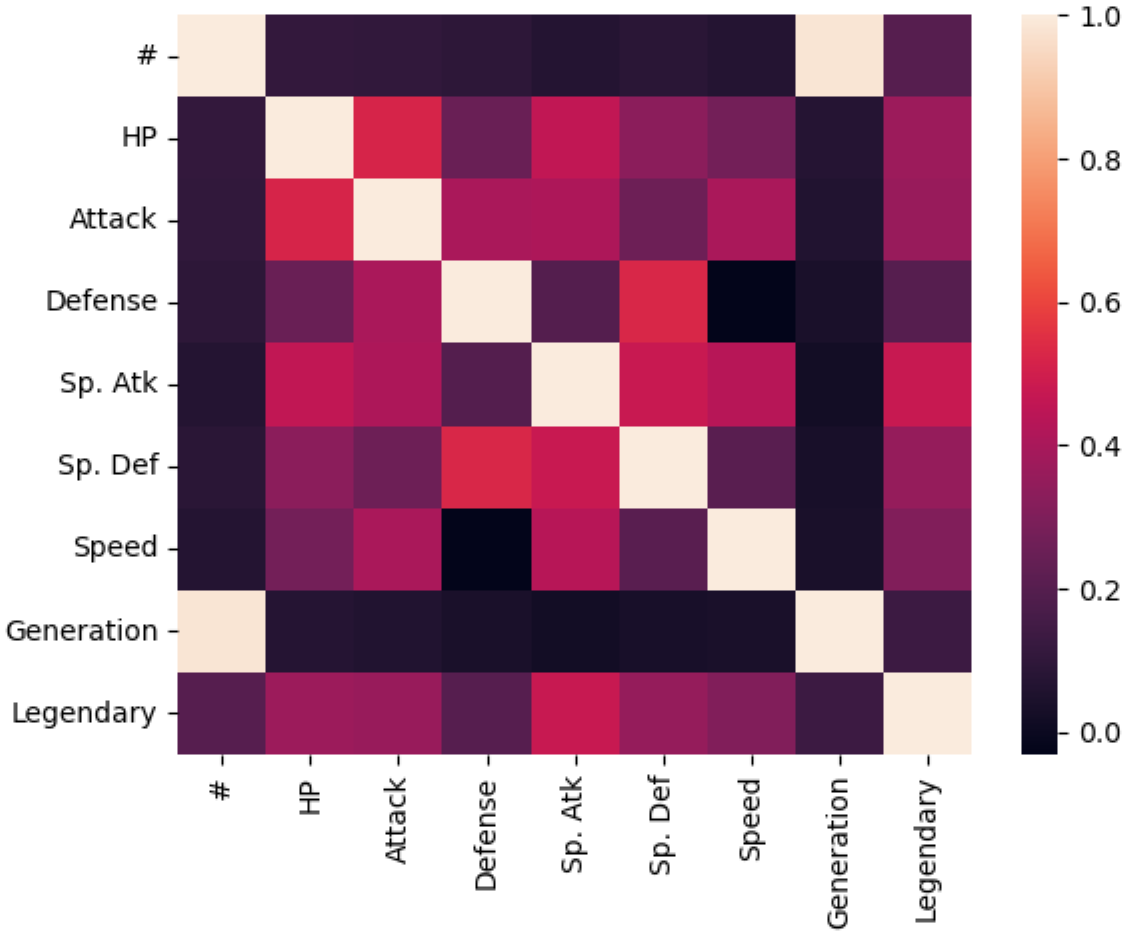
```
In [28]: print( pokeman_Non_na.corr())
```

```
                   #        HP    Attack   Defense   Sp. Atk   Sp. Def  \
#           1.000000  0.109955  0.105918  0.090547  0.065748  0.084785
HP          0.109955  1.000000  0.518707  0.248920  0.456355  0.328665
Attack      0.105918  0.518707  1.000000  0.401001  0.408570  0.257964
Defense     0.090547  0.248920  0.401001  1.000000  0.196778  0.528286
Sp. Atk     0.065748  0.456355  0.408570  0.196778  1.000000  0.480027
Sp. Def     0.084785  0.328665  0.257964  0.528286  0.480027  1.000000
Speed       0.068507  0.271853  0.403546 -0.030993  0.435450  0.208259
Generation  0.983625  0.069728  0.058433  0.040160  0.018942  0.036356
Legendary   0.201582  0.368597  0.359763  0.199898  0.474865  0.354490

               Speed  Generation  Legendary
#           0.068507    0.983625   0.201582
HP          0.271853    0.069728   0.368597
Attack      0.403546    0.058433   0.359763
Defense    -0.030993    0.040160   0.199898
Sp. Atk     0.435450    0.018942   0.474865
Sp. Def     0.208259    0.036356   0.354490
Speed       1.000000    0.039029   0.305780
Generation  0.039029    1.000000   0.130808
Legendary   0.305780    0.130808   1.000000
```

```
In [29]: sns.heatmap(pokeman_Non_na.corr())
```

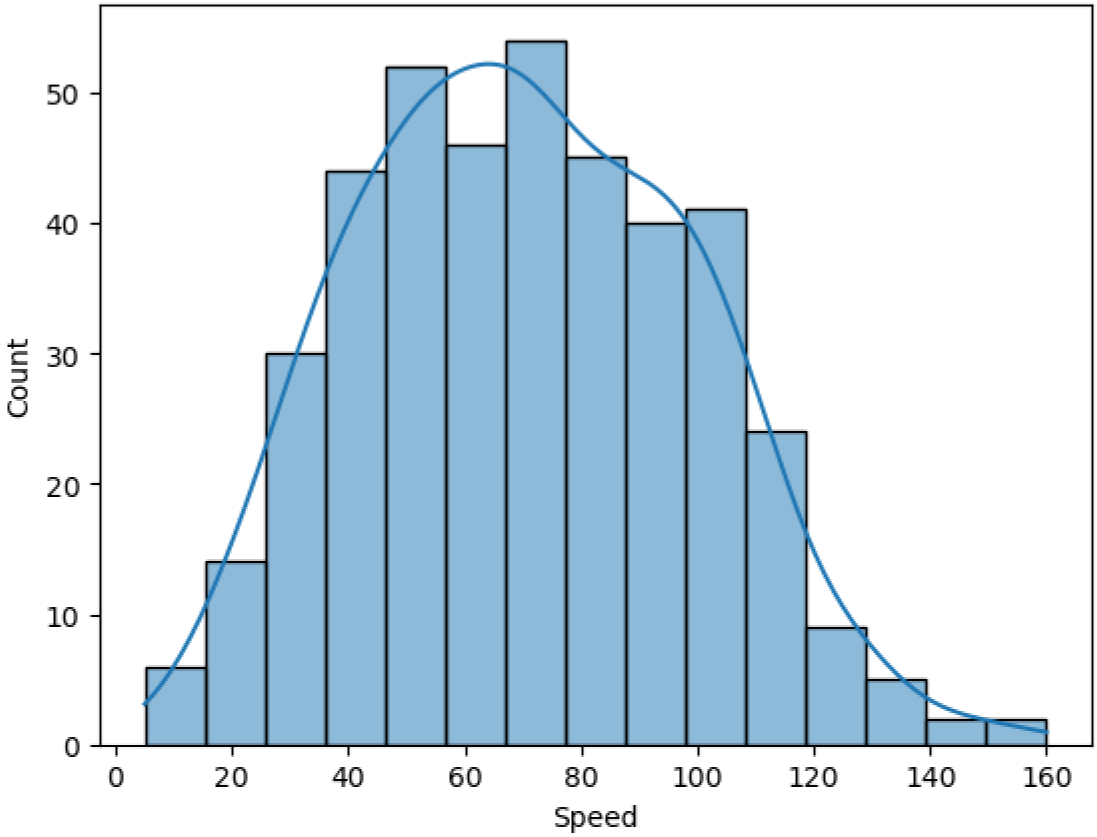
```
Out[29]: <AxesSubplot:>
```



the above heatmap shows the correlation relationship between the characteristic of the pokeman characters

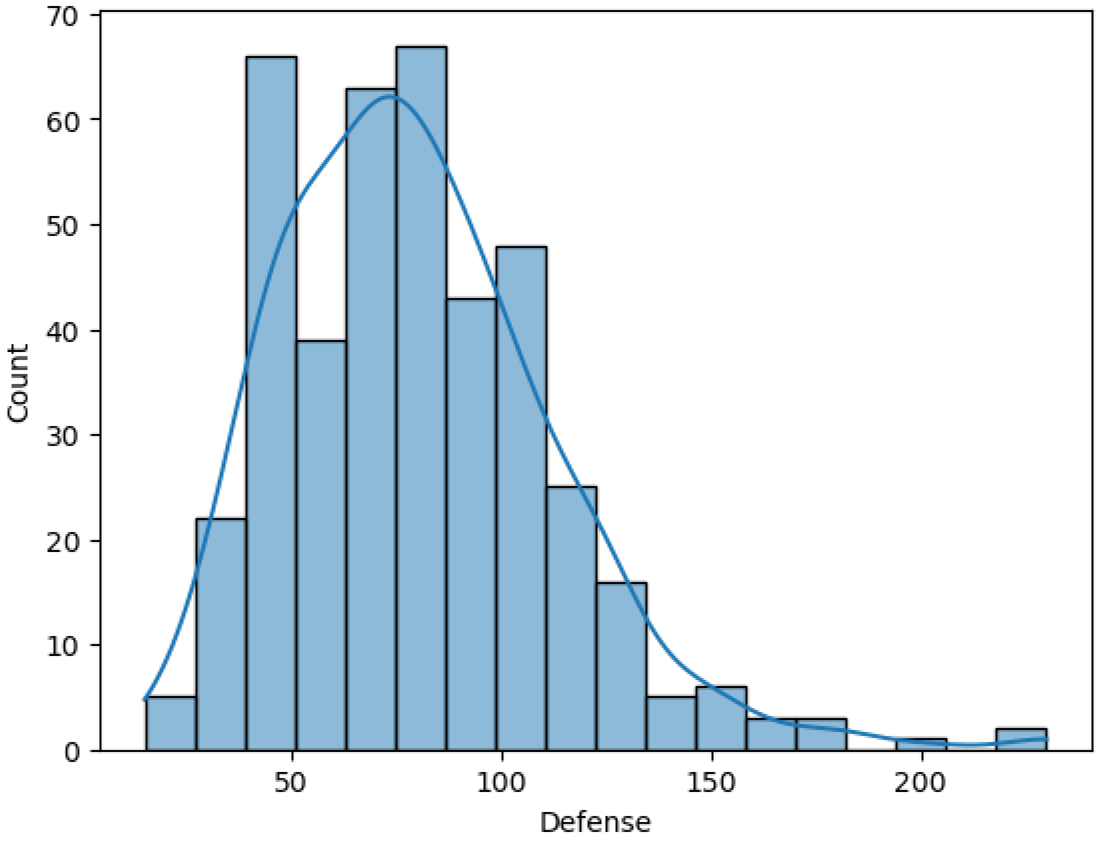
```
In [34]: sns.histplot(data=pokeman_Non_na, x='Speed', kde=True)
```

```
Out[34]: <AxesSubplot:xlabel='Speed', ylabel='Count'>
```



```
In [33]: sns.histplot(data=pokeman_Non_na, x='Defense', kde=True)
```

```
Out[33]: <AxesSubplot:xlabel='Defense', ylabel='Count'>
```



```
In [38]: #sns.pairplot(data=pokeman_Non_na)
         data.drop('Legendary', axis=1)
```

Out[38]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1
6	6	Charizard	Fire	Flying	78	84	78	109	85	100	1
...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	50	100	150	100	150	50	6
796	719	DiancieMega Diancie	Rock	Fairy	50	160	110	160	110	110	6
797	720	HoopaHoopa Confined	Psychic	Ghost	80	110	60	150	130	70	6
798	720	HoopaHoopa Unbound	Psychic	Dark	80	160	60	170	130	80	6
799	721	Volcanion	Fire	Water	80	110	120	130	90	70	6

414 rows × 11 columns

In [40]:

```
data = pokeman
```

In [47]:

```
df=pokeman.drop(['Legendary'], axis=1)
```

In [48]:

```
df
```

Out[48]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1
...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	50	100	150	100	150	50	6
796	719	DiancieMega Diancie	Rock	Fairy	50	160	110	160	110	110	6
797	720	HoopaHoopa Confined	Psychic	Ghost	80	110	60	150	130	70	6
798	720	HoopaHoopa Unbound	Psychic	Dark	80	160	60	170	130	80	6
799	721	Volcanion	Fire	Water	80	110	120	130	90	70	6

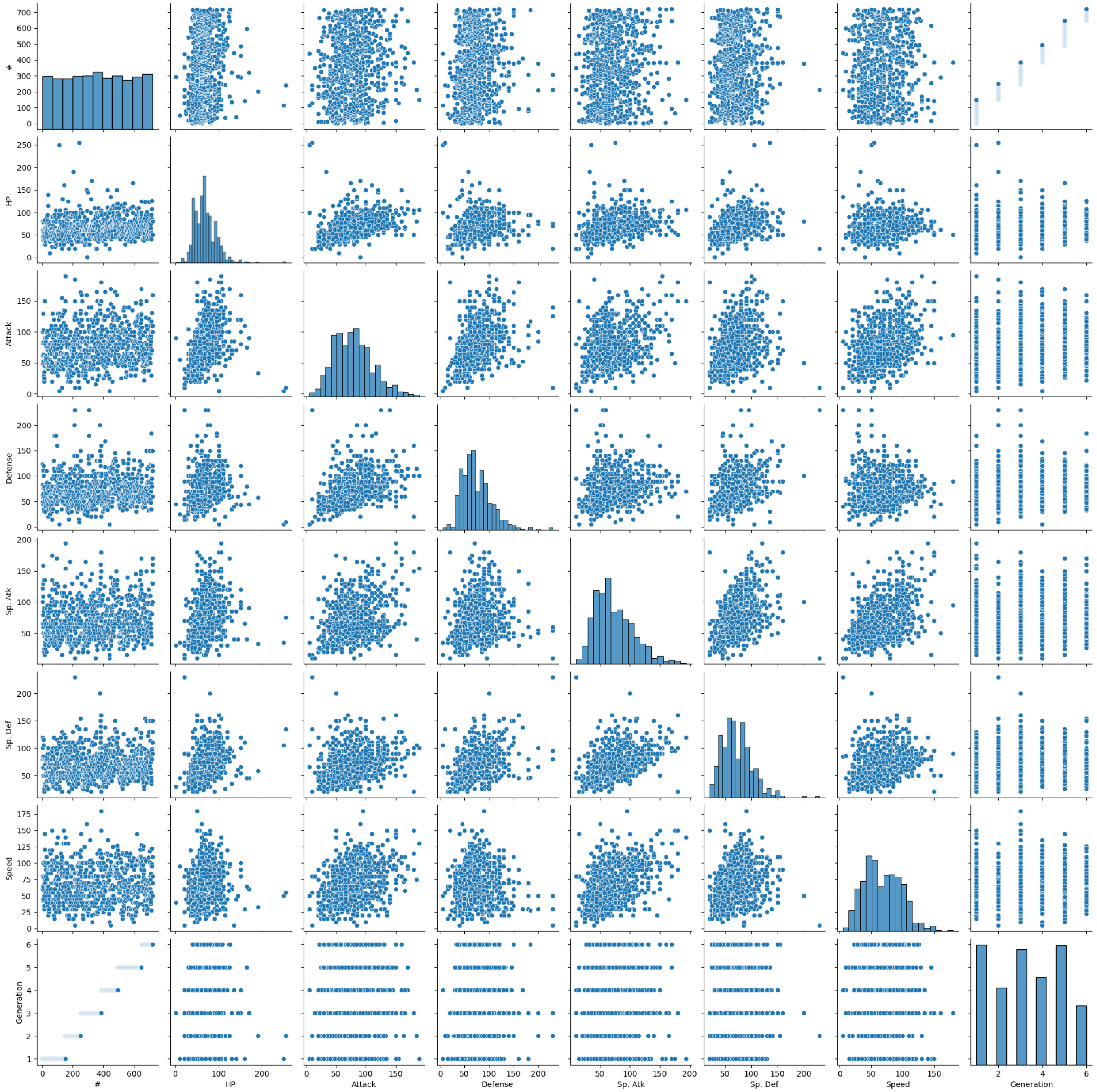
800 rows × 11 columns

In [49]:

```
sns.pairplot(data=df)
```

Out[49]:

```
<seaborn.axisgrid.PairGrid at 0x15110a8ba00>
```


#
0
100
200
300
400
500
600
700
HP
0
50
100
150
200
250
Attack
0
50
100
150
Defense
0
50
100
150
200
Sp. Atk
50
100
150
200
Sp. Def
50
100
150
200
Speed
0
25
50
75
100
125
150
175
Generation
1
2
3
4
5
6
#
0
200
400
600
HP
0
100
200
Attack
0
50
100
150
Defense
0
50
100
150
200
Sp. Atk
50
100
150
200
Sp. Def
50
100
150
200
Speed
0
50
100
150
Generation
2
4
6

In [ ]: