ANALYSIS POKEMON DATA

Data Science Project

Nabilah Fayza Amani

ibimbing

POKEMON.CSV

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	#	Name	Type 1	Typ...	To...	HP	Atta...	Defen...	Sp. ...	Sp. ...	Spe...	Generati...	Legend.
2	1	Bulbasaur	Grass	Pois...	318	45	49	49	65	65	45	1	False
3	2	Ivysaur	Grass	Pois...	405	60	62	63	80	80	60	1	False
4	3	Venusaur	Grass	Pois...	525	80	82	83	100	100	80	1	False
5	3	VenusaurMega Venusaur	Grass	Pois...	625	80	100	123	122	120	80	1	False
6	4	Charmander	Fire		309	39	52	43	60	50	65	1	False
7	5	Charmeleon	Fire		405	58	64	58	80	65	80	1	False
8	6	Charizard	Fire	Flying	534	78	84	78	109	85	100	1	False
9	6	CharizardMega Charizard...	Fire	Drag...	634	78	130	111	130	85	100	1	False
10	6	CharizardMega Charizard Y	Fire	Flying	634	78	104	78	159	115	100	1	False
11	7	Squirtle	Water		314	44	48	65	50	64	43	1	False
12	8	Wartortle	Water		405	59	63	80	65	80	58	1	False
13	9	Blastoise	Water		530	79	83	100	85	105	78	1	False
14	9	BlastoiseMega Blastoise	Water		630	79	103	120	135	115	78	1	False
15	10	Caterpie	Bug		195	45	30	35	20	20	45	1	False
16	11	Metapod	Bug		205	50	20	55	25	25	30	1	False

LOAD DATA

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

poda = pd.read_csv('Pokemon.csv')
poda
```

- Import 3 Libraries such as pandas, seaborn, multi..
- CSV file named “Pokemon.csv“ Readable and It has been saved using pandas

	#	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	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	True
796	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	True

800 rows × 13 columns

DISPLAY INFO DATA

```
poda.info()
✓ 0.1s
```

- From the information that has been included there are 15 rows and 800 columns, but there are 3 different types As with “type 1“, “type 2“ and “Legendary“ is a string

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 16 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    object
 13  Total Defense    800 non-null    int64
 14  Total Strength   800 non-null    int64
 15  Total Stats      800 non-null    int64
dtypes: int64(12), object(4)
memory usage: 100.1+ KB
```

DESCRIBE DATA

```
poda.describe()
✓ 0.1s
```

	#	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Total Defense	S
count	800.000000	800.00000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.00000	800.000000	8
mean	362.813750	435.10250	69.258750	79.001250	73.842500	72.820000	71.902500	68.277500	3.32375	145.745000	4
std	208.343798	119.96304	25.534669	32.457366	31.183501	32.722294	27.828916	29.060474	1.66129	51.315827	1
min	1.000000	180.00000	1.000000	5.000000	5.000000	10.000000	20.000000	5.000000	1.00000	35.000000	1
25%	184.750000	330.00000	50.000000	55.000000	50.000000	49.750000	50.000000	45.000000	2.00000	109.750000	3
50%	364.500000	450.00000	65.000000	75.000000	70.000000	65.000000	70.000000	65.000000	3.00000	140.000000	4
75%	539.250000	515.00000	80.000000	100.000000	90.000000	95.000000	90.000000	90.000000	5.00000	180.000000	5
max	721.000000	780.00000	255.000000	190.000000	230.000000	194.000000	230.000000	180.000000	6.00000	460.000000	7

WHAT ARE THE NAMES OF POKEMON IN THE 5TH GENERATION

There are around 165 pokemon in the 5th generation

```
Pokémon in Generation 5:
553                  Victini
554                    Snivy
555                  Servine
556                 Serperior
557                    Tepig
                ...
713        KeldeoOrdinary Forme
714        KeldeoResolute Forme
715          MeloettaAria Forme
716     MeloettaPirouette Forme
717                  Genesect
Name: Name, Length: 165, dtype: object
```

```
# Filter Pokémon in Generation 5
gen5_pokemon = poda[poda['Generation'] == 5]

# Get the names of Pokémon in Generation 5
gen5_pokemon_names = gen5_pokemon['Name']

# Print the names of Pokémon in Generation 5
print("Pokémon in Generation 5:")
print(gen5_pokemon_names)
```

✓ 0.0s

WHAT ARE THE STRONGEST AND WEAKEST POKEMON

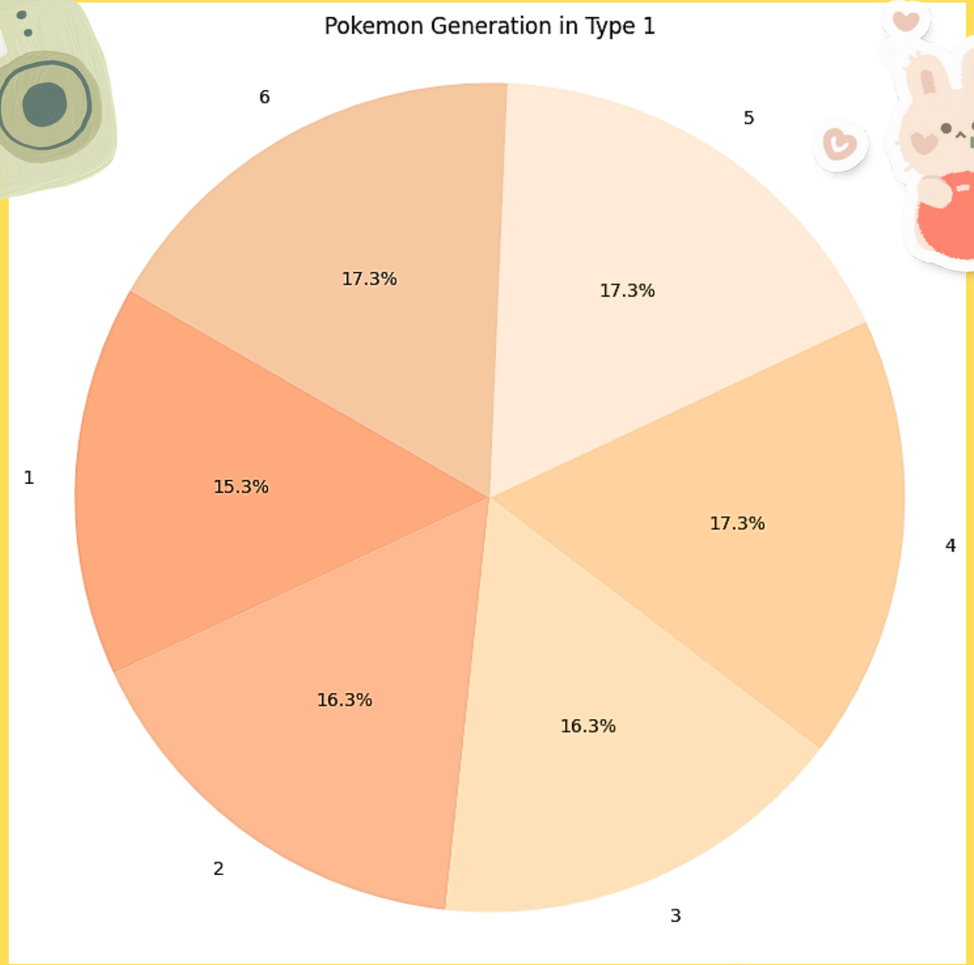
```python
# Define a function to calculate the total stats of a Pokémon
def calculate_total_stats(row):
    return row['HP'] + row['Attack'] + row['Defense'] + row['Sp. Atk'] + row['Sp. Def'] + row['Speed']

# Calculate the total stats for each Pokémon
poda['Total Stats'] = poda.apply(calculate_total_stats, axis=1)

# Find the weakest Pokémon ( lowest total stats )
weakest_pokemon = poda.loc[poda['Total Stats'].idxmin()]
print("Weakest Pokémon:")
print(weakest_pokemon[['Name', 'Type 1', 'Type 2', 'Total Stats']])

# Find the strongest Pokémon ( highest total stats )
strongest_pokemon = poda.loc[poda['Total Stats'].idxmax()]
print("\nStrongest Pokémon:")
print(strongest_pokemon[['Name', 'Type 1', 'Type 2', 'Total Stats']])
```

✓ 0.0s

```
Weakest Pokémon:
Name            Sunkern
Type 1            Grass
Type 2              NaN
Total Stats         180
Name: 206, dtype: object

Strongest Pokémon:
Name            MewtwoMega Mewtwo X
Type 1                      Psychic
Type 2                     Fighting
Total Stats                     780
Name: 163, dtype: object
```

The weakest Pokemon is occupied by sunkern with 180 while the strongest is occupied by skill 780

HOW MANY HOW MANY OF ALL GENERATIONS OF POKEMON ARE TYPE 1?

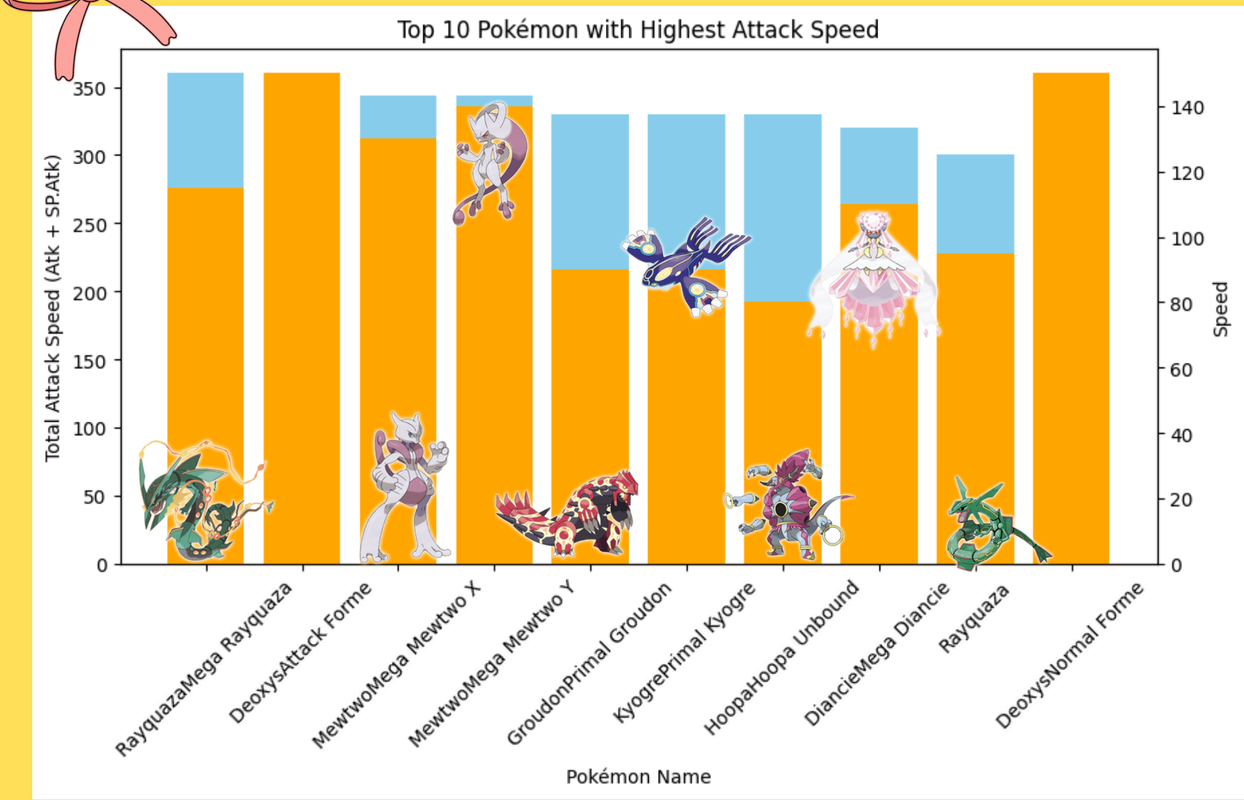



```
#filter data only for type 1 & defining colors
total_per_generation = poda.groupby(['Generation','Type 1'])['#'].count()
generation_counts = total_per_generation.groupby('Generation').count()

pastel = ['#EFB495', '#FFBE98', '#FFE7C1', '#FFD8A9', '#FDEEDC', '#ECCDB4']

#Making Pie Chart
plt.figure(figsize=(9,9))
plt.pie(generation_counts,labels = generation_counts.index, autopct='%1.1f%%', startangle=150, colors=pastel)

#add title
plt.title('Pokemon Generation in Type 1')
plt.axis('equal')

plt.show()
```

Of the six generations, only the first generation has a population about 15.3% less than the others

10 POKEMON THAT HAVE SPEED IN ATTACK

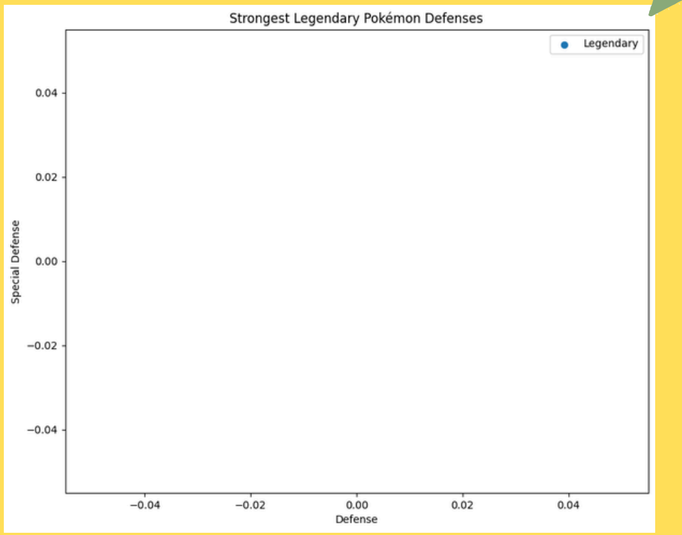
```
# Calculate the total attack speed by adding ATK and SP.ATK
poda['Total Attack Speed'] = poda['Attack'] + poda['Sp. Atk']

# Sort the data by Total Attack Speed in descending order
poda_sorted = poda.sort_values(by='Total Attack Speed', ascending=False)

# Select the top 10 Pokémon with the highest attack speed
top_10_pokemon = poda_sorted.head(10)

# Create a bar chart to visualize the results
plt.figure(figsize=(10, 5))
plt.bar(top_10_pokemon['Name'], top_10_pokemon['Total Attack Speed'], color='skyblue')
plt.xlabel('Pokémon Name')
plt.ylabel('Total Attack Speed (Atk + SP.Atk)')
plt.title('Top 10 Pokémon with Highest Attack Speed')
plt.xticks(rotation=45)

# Add a secondary axis for Speed
plt.twinx()
plt.bar(top_10_pokemon['Name'], top_10_pokemon['Speed'], color='orange')
plt.ylabel('Speed')

plt.show()
```

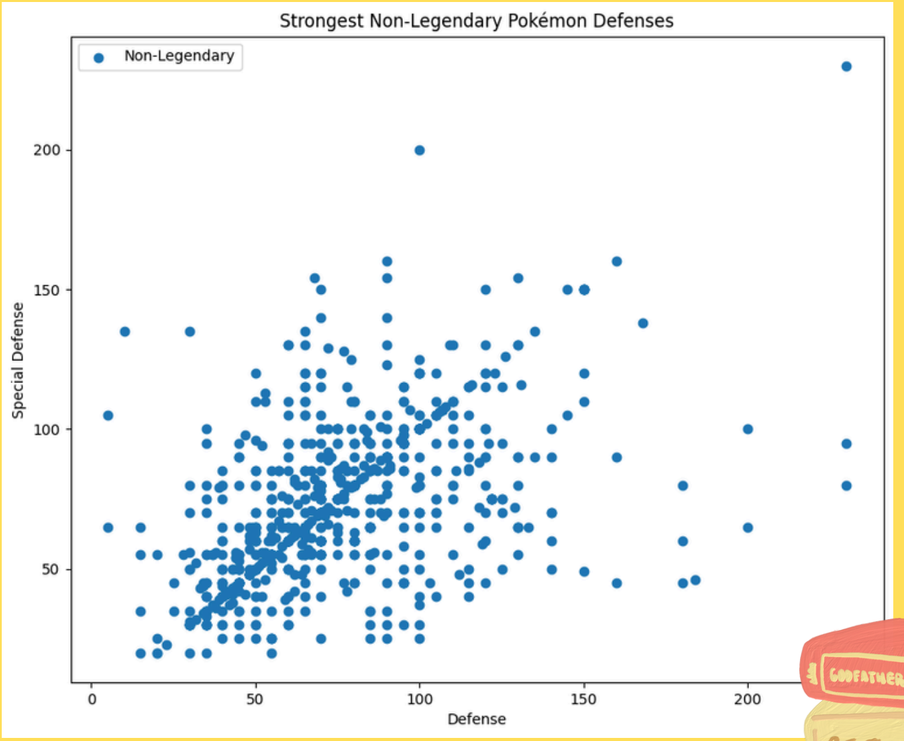


It can be seen that there are 2 Pokemon that have quite a lot of strength in attacking enemies Namely occupied by MewtwoMega Mewtwo X and Y, while the others have some speed that can help them to dodge.

CATEGORY LEGENDARY OR NON-LEGENDARY CATEGORIES THAT HAVE STRENGTH IN DEFENDING AGAINST ENEMY ATTACKS

```python
# Calculate the total defense by adding Defense and Special Defense
poda['Total Defense'] = poda['Defense'] + poda['Sp. Def']

# Create a new column to categorize Pokémon as Legendary or Non-Legendary
poda['Legendary'] = poda['Legendary'].apply(lambda x: 'Legendary' if x == True else 'Non-Legendary')

# Create a scatter plot for Legendary Pokémon
plt.figure(figsize=(10, 8))
plt.scatter(poda[poda['Legendary'] == 'Legendary']['Defense'], poda[poda['Legendary'] == 'Legendary']['Sp. Def'], labe
plt.xlabel('Defense')
plt.ylabel('Special Defense')
plt.title('Strongest Legendary Pokémon Defenses')
plt.legend()

# Create a scatter plot for Non-Legendary Pokémon
plt.figure(figsize=(10, 8))
plt.scatter(poda[poda['Legendary'] == 'Non-Legendary']['Defense'], poda[poda['Legendary'] == 'Non-Legendary']['Sp. Def
plt.xlabel('Defense')
plt.ylabel('Special Defense')
plt.title('Strongest Non-Legendary Pokémon Defenses')
plt.legend()

plt.show()
```





Can be concluded that the Legendary category cannot provide strong defense compared to non-legendary