Meus Melhores Pokémons

Contextualizando os Dados

Para realizar esse projeto, eu criei um conjunto de dados chamado "Meus_Pokémons".

Todos os Pokémons listados na análise foram retirados da minha conta. Procurei listas apenas os melhores "monstrinhos" e sem repetição, ou seja, apenas um registro de cada.

A base de dados possui as seguintes colunas:

Pokémon: Nome do Pokémon capturado;

Detalhe: Informa se o Pokémon capturado é Normal, Shiny, Sombrio ou Sortudo;

À Evoluir: Informa se o Pokémon capturado terá outra evolução;

Ataque: Informa a força de ataque do Pokémon capturado;

Defesa: Informa a defesa do Pokémon capturado;

Ponto de Saúde (PS): Informa a saúde do Pokémon capturado;

Total: E a soma do Ataque, Defesa e PS do Pokémon capturado;

Ponto de Combate (CP): Informa o ponto de combate do Pokémon capturado;

Porcentagem: Informa o percenual do Pokémon capturado;

Estrela: Informa a quantidade de estrelas do Pokémon capturado;

Peso: Informa o peso do Pokémon capturado;

Primário: Informa qual o tipo primário do Pokémon capturado;

Secundário: Informa o tipo secundário do Pokémon capturado;

Atura: Informa a altura do Pokémon capturado;

Sexo: Informa o sexo do Pokémon capturado;

Geração: Informa a geração do Pokémon capturado;

Imagem: Informa a imagem do Pokémon capturado.

Ao todo, são 384 registros!

Consultas Realizadas para o Projeto

A imagens dos Pokémons foram extraídos de: https://www.pokemon.com/br

A documentação da API utilizada para realização de análise está disponível em: https://pogoapi.net/documentation/

In [2]:
```
import pandas as pd
import requests
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
# Carregando a planilha

caminho_do_arquivo = 'Meus_Pokémons.xlsx'
pokemons_df = pd.read_excel(caminho_do_arquivo)
```

In [4]:
```
# Visualizando as primeiras linhas da planilha

pokemons_df.head()
```

Out[4]:

	Pokemon	Detalhe	À Evoluir	Ataque	Defesa	Ponto de Saúde (PS)	Total	Ponto de Combate (CP)	Porcentagem	Estrelas	Peso	Primário	Secundário	Altura	Sexo	Geração	
0	Blissey	Normal	Não	15	15	15	45	2757	100.0	3	53.73	Normal	Normal	1.69	Feminino	2	d

	Pokemon	Detalhe	À Evoluir	Ataque	Defesa	Ponto de Saúde (PS)	Total	Ponto de Combate (CP)	Porcentagem	Estrelas	Peso	Primário	Secundário	Altura	Sexo	Geração	
1	Exploud	Normal	Não	15	15	15	45	2347	100.0	3	84.00	Normal	Normal	1.50	Feminino	3	d
2	Lopunny	Normal	Não	15	15	15	45	2059	100.0	3	33.30	Normal	Normal	1.20	Feminino	4	d
3	Perrserker	Normal	Não	15	15	15	45	2401	100.0	3	28.00	Aço	Aço	0.80	Feminino	8	d
4	Porygon-Z	Normal	Não	15	15	15	45	3266	100.0	3	36.50	Normal	Normal	0.80	Indefinido	1	d

In [5]:
```
# Verificando a quantidade de linhas e colunas da planilha

print("Quantidade de linhas e colunas da planilha:")
print()
print(pokemons_df.shape)
```

```
Quantidade de linhas e colunas da planilha:

(384, 17)
```

In [6]:
```
# Verificando nome das colunas da planilha

print("Colunas na planilha:")
print()
print(pokemons_df.columns)
```

```
Colunas na planilha:

Index(['Pokemon', 'Detalhe', 'À Evoluir', 'Ataque', 'Defesa',
       'Ponto de Saúde (PS)', 'Total', 'Ponto de Combate (CP)', 'Porcentagem',
       'Estrelas', 'Peso', 'Primário', 'Secundário', 'Altura', 'Sexo',
       'Geração', 'Imagem'],
      dtype='object')
```

In [7]:
```
# Função para obter dados do CP Máximo da API

def get_max_cp_data():
```

```
    url = "https://pogoapi.net/api/v1/pokemon_max_cp.json"
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    else:
        raise Exception(f"Erro ao acessar a API: {response.status_code}")

# Obtendo dados de CP Máximo

max_cp_data = get_max_cp_data()

# Convertendo os dados de CP Máximo em um DataFrame

max_cp_df = pd.DataFrame(max_cp_data)

# Visualizando as primeiras linhas do DataFrame para verificar os dados

max_cp_df.head()
```

Out[7]:

	form	max_cp	pokemon_id	pokemon_name
0	Fall_2019	1275	1	Bulbasaur
1	Normal	1275	1	Bulbasaur
2	Normal	1943	2	Ivysaur
3	Copy_2019	3112	3	Venusaur
4	Normal	3112	3	Venusaur

In [8]:

```
# Verificando a quantidade de linhas e colunas da API

print("Quantidade de linhas e colunas da API:")
print()
print(max_cp_df.shape)
```

```
Quantidade de linhas e colunas da API:

(1328, 4)
```

In [9]:

```
# Verificando nomes das colunas do Dataframe da API
```

```
print("Colunas no DataFrame:")
print()
print(max_cp_df.columns)
```

```
Colunas no DataFrame:

Index(['form', 'max_cp', 'pokemon_id', 'pokemon_name'], dtype='object')
```

In [10]:

```
# Realizando a integração dos dados da planilha com o DataFrame da API
# Renomeando colunas para facilitar a combinação

max_cp_df.rename(columns = {'pokemon_name': 'Pokemon', 'max_cp': 'CP_Máximo'}, inplace = True)
```

In [11]:

```
# Integrando os dados de CP Máximo na planilha

pokemons_df = pd.merge(pokemons_df, max_cp_df[['Pokemon', 'CP_Máximo']], on = 'Pokemon', how = 'left')

# Visualizando as primeiras linhas para verificar a integração

print(pokemons_df.head())
```

```
      Pokemon Detalhe À Evoluir  Ataque  Defesa  Ponto de Saúde (PS)  Total  \
0     Blissey  Normal       Não      15      15                   15     45
1     Exploud  Normal       Não      15      15                   15     45
2     Lopunny  Normal       Não      15      15                   15     45
3  Perrserker  Normal       Não      15      15                   15     45
4   Porygon-Z  Normal       Não      15      15                   15     45

   Ponto de Combate (CP)  Porcentagem  Estrelas   Peso Primário Secundário  \
0                   2757        100.0         3  53.73   Normal     Normal
1                   2347        100.0         3  84.00   Normal     Normal
2                   2059        100.0         3  33.30   Normal     Normal
3                   2401        100.0         3  28.00      Aço        Aço
4                   3266        100.0         3  36.50   Normal     Normal

   Altura        Sexo  Geração  \
0    1.69    Feminino        2
1    1.50    Feminino        3
2    1.20    Feminino        4
3    0.80    Feminino        8
4    0.80  Indefinido        1

                                              Imagem  CP_Máximo
0  ...     3155.0
```

```
1  ...     2685.0
2  ...     2356.0
3  ...     2729.0
4  ...     3737.0
```

In [12]:
```
# Verificando a quantidade de linhas e colunas da planilha

print("Quantidade de linhas e colunas da planilha:")
print()
print(pokemons_df.shape)
```

```
Quantidade de linhas e colunas da planilha:

(507, 18)
```

In [13]:
```
# Verificando duplicatas

duplicatas = pokemons_df['Pokemon'].value_counts()
```

In [14]:
```
# Filtrando apenas os pokémons que aparecem mais de uma vez

duplicatas = duplicatas[duplicatas > 1]
```

In [15]:
```
# Exibindo os pokémons duplicados e a quantidade de vezes que eles aparecem

print("Pokémons duplicados e a quantidade de vezes que aparecem:")
print()
print(duplicatas)
```

```
Pokémons duplicados e a quantidade de vezes que aparecem:

Pokemon
Pikachu       36
Genesect       5
Castform       4
Darmanitan     4
Deoxys         4
              ..
Vulpix         2
Sandslash      2
Raticate       2
```

```
Thundurus      2
Suicune        2
Name: count, Length: 69, dtype: int64
```

In [16]:
```
# Convertendo para DataFrame para visualização mais clara

duplicatas_df = duplicatas.reset_index()
duplicatas_df.columns = ['Pokemon', 'Quantidade']
print()
print(duplicatas_df)
```

```
       Pokemon  Quantidade
0      Pikachu          36
1     Genesect           5
2     Castform           4
3   Darmanitan           4
4       Deoxys           4
..         ...         ...
64      Vulpix           2
65   Sandslash           2
66    Raticate           2
67   Thundurus           2
68     Suicune           2

[69 rows x 2 columns]
```

In [17]:
```
# Removendo duplicatas, mantendo apenas o primeiro registro de cada Pokémon

pokemons_df_sem_duplicatas = pokemons_df.drop_duplicates(subset='Pokemon', keep = 'first')
```

In [18]:
```
# Verificando a quantidade de linhas e colunas da planilha após a exclusão dos pokémons repetidos

print("Quantidade de linhas e colunas da planilha:")
print()
print(pokemons_df_sem_duplicatas.shape)
```

```
Quantidade de linhas e colunas da planilha:

(384, 18)
```

In [19]:
```
# Verificando a quantidade de valores nulos por coluna
```

```
print("Dados nulos por coluna:")
print()
print(pokemons_df_sem_duplicatas.isnull().sum())
```

```
Dados nulos por coluna:

Pokemon                  0
Detalhe                  0
À Evoluir                0
Ataque                   0
Defesa                   0
Ponto de Saúde (PS)      0
Total                    0
Ponto de Combate (CP)    0
Porcentagem              0
Estrelas                 0
Peso                     0
Primário                 0
Secundário               0
Altura                   0
Sexo                     0
Geração                  0
Imagem                   0
CP_Máximo                3
dtype: int64
```

In [20]:

```
# Verificando a quantidade de valores nulos por linha

valores_nulos_linhas = pokemons_df_sem_duplicatas.isnull().sum(axis=1)
```

In [21]:

```
# Contando a quantidade de linhas que possuem pelo menos um valor nulo

linhas_com_valores_nulos = valores_nulos_linhas[valores_nulos_linhas > 0].shape[0]

print(f"Quantidade de linhas com pelo menos um valor nulo: {linhas_com_valores_nulos}")
```

```
Quantidade de linhas com pelo menos um valor nulo: 3
```

In [22]:

```
# Verificando a quantidade total de valores nulos

total_valores_nulos = pokemons_df_sem_duplicatas.isnull().sum().sum()

print(f"Quantidade total de valores nulos: {total_valores_nulos}")
```

```
Quantidade total de valores nulos: 3
```

In [23]:
```
# Verificando quais Pokémons têm valores nulos na coluna CP_Máximo

pokemons_com_cp_maximo_nulo = pokemons_df_sem_duplicatas[pokemons_df_sem_duplicatas['CP_Máximo'].isnull()]

print("Pokémons sem informações de CP Máximo:")
print(pokemons_com_cp_maximo_nulo[['Pokemon', 'Ponto de Combate (CP)', 'CP_Máximo']])
```

```
Pokémons sem informações de CP Máximo:
      Pokemon  Ponto de Combate (CP)  CP_Máximo
40   Farfetch                    946        NaN
52   Sirfetch                   1691        NaN
160   Nidoran                    335        NaN
```

In [24]:
```
# Renomeando o DataFrame

pokemons_df_atualizado = pokemons_df_sem_duplicatas.copy()
```

In [25]:
```
# Atualizando CP Máximo manualmente

pokemons_df_atualizado.loc[pokemons_df_atualizado['Pokemon'] == 'Farfetch', 'CP_Máximo'] = 1414
pokemons_df_atualizado.loc[pokemons_df_atualizado['Pokemon'] == 'Sirfetch', 'CP_Máximo'] = 3415
pokemons_df_atualizado.loc[pokemons_df_atualizado['Pokemon'] == 'Nidoran', 'CP_Máximo'] = 933
```

In [26]:
```
# Verificando se ainda há valores nulos

print("Dados nulos por coluna após atualização manual:")
print()
print(pokemons_df_atualizado.isnull().sum())
```

```
Dados nulos por coluna após atualização manual:

Pokemon                 0
Detalhe                 0
À Evoluir               0
Ataque                  0
Defesa                  0
Ponto de Saúde (PS)     0
Total                   0
```

```
Ponto de Combate (CP)    0
Porcentagem              0
Estrelas                 0
Peso                     0
Primário                 0
Secundário               0
Altura                   0
Sexo                     0
Geração                  0
Imagem                   0
CP_Máximo                0
dtype: int64
```

In [27]:

```
# Verificando a quantidade total de valores nulos após atualização

total_valores_nulos_atualizado = pokemons_df_atualizado.isnull().sum().sum()

print(f"Quantidade total de valores nulos após atualização: {total_valores_nulos_atualizado}")
```

```
Quantidade total de valores nulos após atualização: 0
```

In [28]:

```
# Calculando a porcentagem do CP atual em relação ao CP Máximo

pokemons_df_atualizado['Porcentagem_CP'] = (pokemons_df_atualizado['Ponto de Combate (CP)'] / pokemons_df_atualizado['CP_Máximo'])

# Ordenando o DataFrame pela porcentagem do CP atual em relação ao CP máximo

pokemons_df_atualizado = pokemons_df_atualizado.sort_values(by='Porcentagem_CP', ascending=False)
```

In [29]:

```
# Visualizando as primeiras linhas do DataFrame ordenado

print(pokemons_df_atualizado.head(10))
```

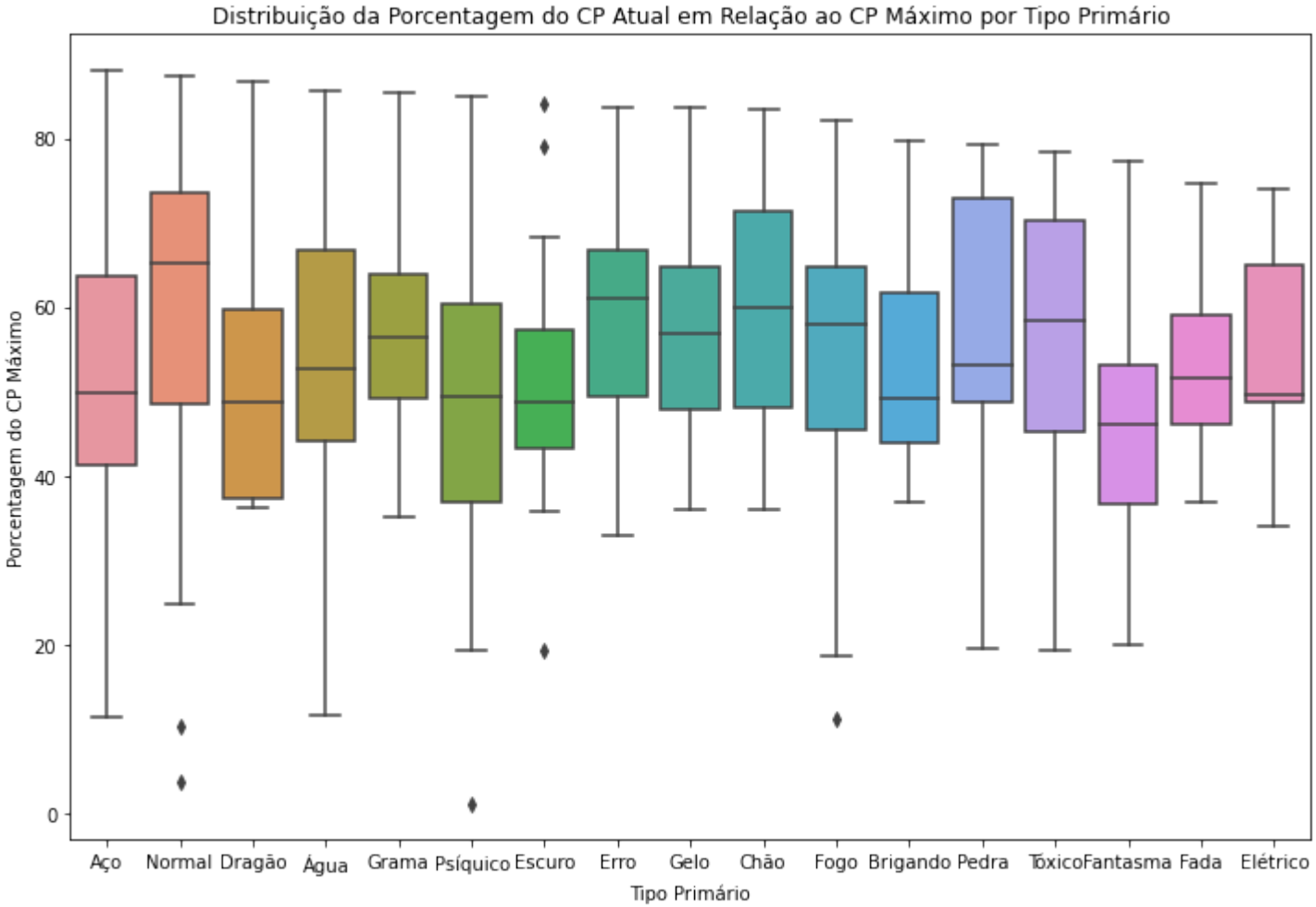
```
        Pokemon Detalhe À Evoluir  Ataque  Defesa  Ponto de Saúde (PS)  Total  \
3    Perrserker  Normal       Não      15      15                   15     45
1       Exploud  Normal       Não      15      15                   15     45
4     Porygon-Z  Normal       Não      15      15                   15     45
2       Lopunny  Normal       Não      15      15                   15     45
0       Blissey  Normal       Não      15      15                   15     45
70    Dragonite  Normal       Não      15      14                   12     41
190    Gyarados  Normal       Não      13      10                   15     38
187   Exeggutor  Normal       Não      12      12                   14     38
291    Alakazam  Normal       Não      15      15                    6     36
```

```
354     Weavile  Normal       Não      10      12                   11     33

     Ponto de Combate (CP)  Porcentagem  Estrelas    Peso  Primário  \
3                     2401   100.000000         3   28.00       Aço
1                     2347   100.000000         3   84.00    Normal
4                     3266   100.000000         3   36.50    Normal
2                     2059   100.000000         3   33.30    Normal
0                     2757   100.000000         3   53.73    Normal
70                    3758    91.111111         3  266.78    Dragão
190                   3322    84.444444         3  235.00      Água
187                   2944    84.444444         3  120.00     Grama
291                   2971    80.000000         2   50.00  Psíquico
354                   2891    73.333333         2   34.00    Escuro

    Secundário  Altura        Sexo  Geração  \
3          Aço    0.80    Feminino        8
1       Normal    1.50    Feminino        3
4       Normal    0.80  Indefinido        1
2       Normal    1.20    Feminino        4
0       Normal    1.69    Feminino        2
70         Vôo    2.30   Masculino        1
190        Vôo    6.50   Masculino        1
187   Psíquico    2.00   Masculino        1
291   Psíquico    1.69   Masculino        1
354       Gelo    1.10    Feminino        4

                                                 Imagem  CP_Máximo  \
3    ...     2729.0
1    ...     2685.0
4    ...     3737.0
2    ...     2356.0
0    ...     3155.0
70   ...     4338.0
190  ...     3879.0
187  ...     3448.0
291  ...     3498.0
354  ...     3438.0

     Porcentagem_CP
3         87.980945
1         87.411546
4         87.396307
2         87.393888
0         87.385103
70        86.629783
190       85.640629
187       85.382831
```

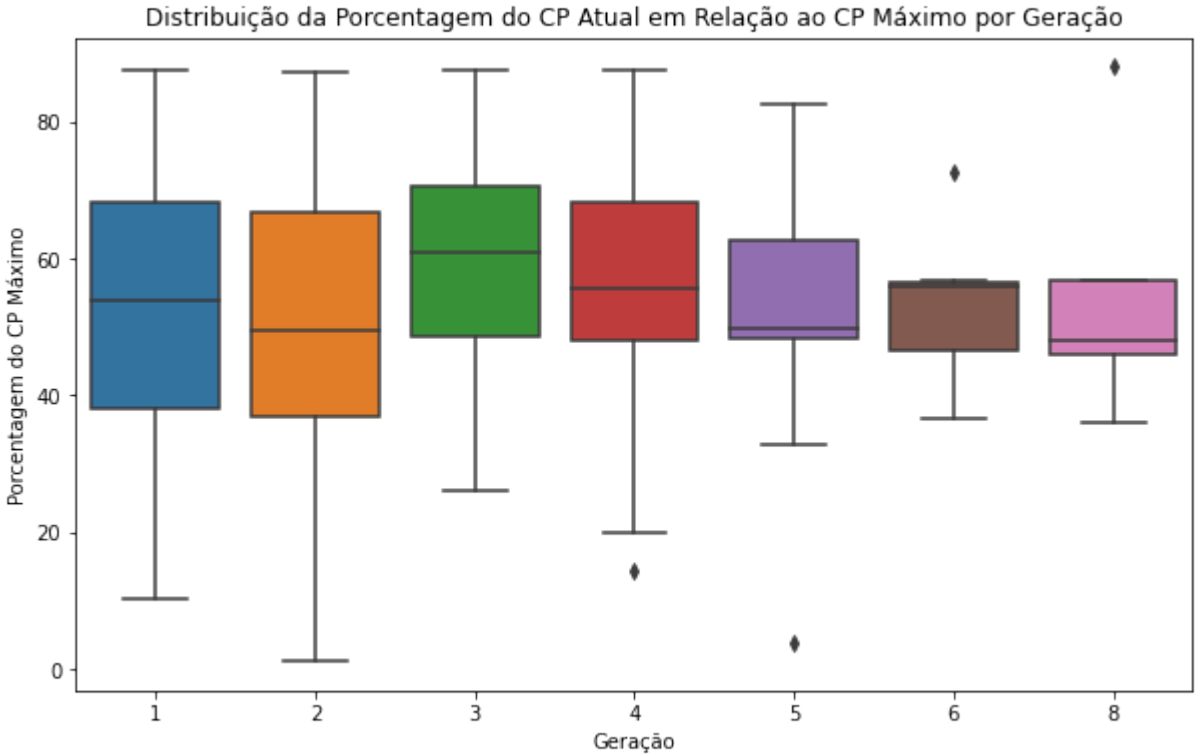
```
291       84.934248
354       84.089587
```

In [30]:

```
# Criando um gráfico de boxplot para a porcentagem do CP atual em relação ao CP máximo por tipo primário

plt.figure(figsize=(12, 8))
sns.boxplot(x='Primário', y='Porcentagem_CP', data=pokemons_df_atualizado)
plt.title('Distribuição da Porcentagem do CP Atual em Relação ao CP Máximo por Tipo Primário')
plt.xlabel('Tipo Primário')
plt.ylabel('Porcentagem do CP Máximo')
plt.show()
```



In [31]:
```
# Criando um gráfico de boxplot para a porcentagem do CP atual em relação ao CP máximo por geração

plt.figure(figsize=(10, 6))
sns.boxplot(x='Geração', y='Porcentagem_CP', data=pokemons_df_atualizado)
plt.title('Distribuição da Porcentagem do CP Atual em Relação ao CP Máximo por Geração')
plt.xlabel('Geração')
plt.ylabel('Porcentagem do CP Máximo')
plt.show()
```



In [32]:
```
# Função para obter dados de estatísticas base da API

def get_base_stats_data():
    url = "https://pogoapi.net/api/v1/pokemon_stats.json"
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
```

```
    else:
        raise Exception(f"Erro ao acessar a API: {response.status_code}")
```

In [33]:
```
# Obtendo dados de estatísticas base

base_stats_data = get_base_stats_data()
```

In [34]:
```
# Convertendo os dados de estatísticas base em um DataFrame

base_stats_df = pd.json_normalize(base_stats_data)
```

In [35]:
```
# Verificando a estrutura dos dados

print(base_stats_df.head())
```

```
   base_attack  base_defense  base_stamina       form  pokemon_id pokemon_name
0          118           111           128  Fall_2019           1    Bulbasaur
1          118           111           128     Normal           1    Bulbasaur
2          151           143           155     Normal           2      Ivysaur
3          198           189           190  Copy_2019           3     Venusaur
4          198           189           190     Normal           3     Venusaur
```

In [36]:
```
# Normalizando os nomes dos Pokémon para garantir a correspondência correta

base_stats_df['pokemon_name'] = base_stats_df['pokemon_name'].str.lower()
pokemons_df_atualizado['Pokemon'] = pokemons_df_atualizado['Pokemon'].str.lower()
```

In [37]:
```
# Integrando os dados da API ao nosso DataFrame existente usando o nome do Pokémon

pokemons_df_atualizado = pokemons_df_atualizado.merge(base_stats_df, left_on='Pokemon', right_on='pokemon_name', how='left')

# Exibindo as primeiras linhas do DataFrame atualizado

print(pokemons_df_atualizado.head())
```

```
      Pokemon Detalhe À Evoluir  Ataque  Defesa  Ponto de Saúde (PS)  Total  \
0  perrserker  Normal       Não      15      15                   15     45
1     exploud  Normal       Não      15      15                   15     45
2   porygon-z  Normal       Não      15      15                   15     45
```

```
3     lopunny  Normal       Não      15      15                  15     45
4     blissey  Normal       Não      15      15                  15     45

   Ponto de Combate (CP)  Porcentagem  Estrelas  ...  Geração  \
0                   2401        100.0         3  ...        8
1                   2347        100.0         3  ...        3
2                   3266        100.0         3  ...        1
3                   2059        100.0         3  ...        4
4                   2757        100.0         3  ...        2

                                             Imagem CP_Máximo  \
0  ...    2729.0
1  ...    2685.0
2  ...    3737.0
3  ...    2356.0
4  ...    3155.0

   Porcentagem_CP base_attack  base_defense base_stamina      form  \
0       87.980945       195.0         162.0        172.0  Galarian
1       87.411546       179.0         137.0        232.0    Normal
2       87.396307       264.0         150.0        198.0    Normal
3       87.393888       156.0         194.0        163.0    Normal
4       87.385103       129.0         169.0        496.0    Normal

   pokemon_id  pokemon_name
0       863.0    perrserker
1       295.0       exploud
2       474.0     porygon-z
3       428.0       lopunny
4       242.0       blissey

[5 rows x 25 columns]
```

In [38]:

```
# Verificando a quantidade de linhas e colunas da planilha após a junção

print("Quantidade de linhas e colunas da planilha:")
print()
print(pokemons_df_atualizado.shape)
```

```
Quantidade de linhas e colunas da planilha:

(507, 25)
```

In [39]:

```
# Verificando se há valores nulos
```

```
print("Dados nulos por coluna após atualização manual:")
print()
print(pokemons_df_atualizado.isnull().sum())
```

```
Dados nulos por coluna após atualização manual:

Pokemon                 0
Detalhe                 0
À Evoluir               0
Ataque                  0
Defesa                  0
Ponto de Saúde (PS)     0
Total                   0
Ponto de Combate (CP)   0
Porcentagem             0
Estrelas                0
Peso                    0
Primário                0
Secundário              0
Altura                  0
Sexo                    0
Geração                 0
Imagem                  0
CP_Máximo               0
Porcentagem_CP          0
base_attack             3
base_defense            3
base_stamina            3
form                    3
pokemon_id              3
pokemon_name            3
dtype: int64
```

In [40]:

```
# Filtrando os Pokémon que têm dados ausentes

pokemons_com_dados_ausentes = pokemons_df_atualizado[pokemons_df_atualizado.isnull().any(axis=1)]
```

In [41]:

```
# Exibindo os Pokémon com dados ausentes

print("Pokémon com dados ausentes:")
print(pokemons_com_dados_ausentes)
```

```
Pokémon com dados ausentes:
      Pokemon Detalhe À Evoluir  Ataque  Defesa  Ponto de Saúde (PS)  Total  \
```

```
125  farfetch  Normal       Sim      15      13                  14     42
270  sirfetch  Normal       Não      15      15                  12     42
466   nidoran  Normal       Sim      12      14                  13     39

     Ponto de Combate (CP)  Porcentagem  Estrelas  ...  Geração  \
125                    946    93.333333         3  ...        1
270                   1691    93.333333         3  ...        1
466                    335    86.666667         3  ...        1

                                                 Imagem CP_Máximo  \
125  ...    1414.0
270  ...    3415.0
466  ...     933.0

     Porcentagem_CP base_attack  base_defense base_stamina  form  pokemon_id  \
125       66.902405         NaN           NaN          NaN   NaN         NaN
270       49.516837         NaN           NaN          NaN   NaN         NaN
466       35.905681         NaN           NaN          NaN   NaN         NaN

     pokemon_name
125           NaN
270           NaN
466           NaN

[3 rows x 25 columns]
```

In [42]:

```python
# Preenchendo os dados ausentes manualmente
# Dados para preencher

dados_ausentes = {
    "farfetch": {
        "base_attack": 124,
        "base_defense": 115,
        "base_stamina": 141,
        "form": "Galarian",
        "pokemon_id": 83,
        "pokemon_name": "farfetch"
    },
    "sirfetch": {
        "base_attack": 248,
        "base_defense": 176,
        "base_stamina": 158,
        "form": "Galarian",
        "pokemon_id": 865,
        "pokemon_name": "sirfetch"
```

```
    },
    "nidoran": {
        "base_attack": 86,
        "base_defense": 89,
        "base_stamina": 120,
        "form": "Normal",
        "pokemon_id": 29,
        "pokemon_name": "nidoran"
    }
}
```

In [43]:

```
# Função para preencher os dados ausentes

def preencher_dados_ausentes(df, dados_ausentes):
    for index, row in df.iterrows():
        if row['Pokemon'] in dados_ausentes:
            df.at[index, 'base_attack'] = dados_ausentes[row['Pokemon']]['base_attack']
            df.at[index, 'base_defense'] = dados_ausentes[row['Pokemon']]['base_defense']
            df.at[index, 'base_stamina'] = dados_ausentes[row['Pokemon']]['base_stamina']
            df.at[index, 'form'] = dados_ausentes[row['Pokemon']]['form']
            df.at[index, 'pokemon_id'] = dados_ausentes[row['Pokemon']]['pokemon_id']
            df.at[index, 'pokemon_name'] = dados_ausentes[row['Pokemon']]['pokemon_name']

# Preenchendo os dados ausentes no DataFrame

preencher_dados_ausentes(pokemons_df_atualizado, dados_ausentes)
```

In [44]:

```
# Verificando novamente se há valores nulos após o preenchimento manual

print("Dados nulos por coluna após preenchimento manual:")
print(pokemons_df_atualizado.isnull().sum())
```

```
Dados nulos por coluna após preenchimento manual:
Pokemon                 0
Detalhe                 0
À Evoluir               0
Ataque                  0
Defesa                  0
Ponto de Saúde (PS)     0
Total                   0
Ponto de Combate (CP)   0
Porcentagem             0
```

```
Estrelas                0
Peso                    0
Primário                0
Secundário              0
Altura                  0
Sexo                    0
Geração                 0
Imagem                  0
CP_Máximo               0
Porcentagem_CP          0
base_attack             0
base_defense            0
base_stamina            0
form                    0
pokemon_id              0
pokemon_name            0
dtype: int64
```

In [45]:
```
# Contando as ocorrências de cada Pokémon no DataFrame

pokemon_counts = pokemons_df_atualizado['pokemon_name'].value_counts()
```

In [46]:
```
# Filtrando apenas os Pokémon que aparecem mais de uma vez

pokemon_duplicados = pokemon_counts[pokemon_counts > 1]
```

In [47]:
```
# Exibindo os Pokémon duplicados e suas quantidades

print(pokemon_duplicados)
```

```
pokemon_name
pikachu      36
genesect      5
sawsbuck      4
castform      4
deoxys        4
             ..
espeon        2
golem         2
exeggutor     2
arcanine      2
cherrim       2
Name: count, Length: 69, dtype: int64
```

In [48]:

```
# Ordenando o DataFrame pelo nome do Pokémon e mantendo apenas o primeiro registro de cada Pokémon duplicado

pokemons_df_unico = pokemons_df_atualizado.drop_duplicates(subset=['pokemon_name'], keep='first')
```

In [49]:

```
# Verificando as primeiras linhas do DataFrame atualizado

print(pokemons_df_unico.head())
```

```
      Pokemon Detalhe À Evoluir  Ataque  Defesa  Ponto de Saúde (PS)  Total  \
0  perrserker  Normal       Não      15      15                   15     45
1     exploud  Normal       Não      15      15                   15     45
2   porygon-z  Normal       Não      15      15                   15     45
3     lopunny  Normal       Não      15      15                   15     45
4     blissey  Normal       Não      15      15                   15     45

   Ponto de Combate (CP)  Porcentagem  Estrelas  ...  Geração  \
0                   2401        100.0         3  ...        8
1                   2347        100.0         3  ...        3
2                   3266        100.0         3  ...        1
3                   2059        100.0         3  ...        4
4                   2757        100.0         3  ...        2

                                             Imagem CP_Máximo  \
0  ...    2729.0
1  ...    2685.0
2  ...    3737.0
3  ...    2356.0
4  ...    3155.0

   Porcentagem_CP base_attack  base_defense base_stamina      form  \
0       87.980945       195.0         162.0        172.0  Galarian
```

```
1       87.411546       179.0         137.0         232.0    Normal
2       87.396307       264.0         150.0         198.0    Normal
3       87.393888       156.0         194.0         163.0    Normal
4       87.385103       129.0         169.0         496.0    Normal

   pokemon_id  pokemon_name
0       863.0    perrserker
1       295.0       exploud
2       474.0     porygon-z
3       428.0       lopunny
4       242.0       blissey

[5 rows x 25 columns]
```

In [50]:

```
# Verificando a quantidade de linhas e colunas da planilha após a exclusão dos Pokémon repetidos

print("Quantidade de linhas e colunas da planilha:")
print(pokemons_df_unico.shape)
```

```
Quantidade de linhas e colunas da planilha:
(384, 25)
```

In [51]:

```
# Salvando o dataFrame atual após todas as modificações

pokemons_df_unico.to_excel('Meus_Pokémons_Atualizados.xlsx', index=False)

print(f'Dados salvos com sucesso em "Meus_Pokémons_Atualizados.xlsx".')
```

```
Dados salvos com sucesso em "Meus_Pokémons_Atualizados.xlsx".
```

In [ ]: