Pokemón Data Analysis

A intenção dessa Análise Exploratória é trabalhar com a Estatística Descritiva. Tem o intuito de descrever e até mesmo coletar algumas informações e curiosidades sobre o dataset de Pokemons. Para isso, além das técnicas para manipulação dos dados, seguirei algumas premissas que me guiarão nessa busca:

1. Qual é a quantidade total de Pokémons neste dataset?
2. Quantos Pokemóns foram lançados em cada geração?
3. Qual é a distribuição dos Pokémons por tipo_1 (predominante)?
4. Qual é a distribuição dos Pokémons por tipo_2 em função do tipo_1?
5. Em cada GERAÇÃO, identificar o Pokémon: i. mais alto (height_m);
   ii. mais pesado (weight_kg);
   iii. maior HP (hp);
   iv. maior ataque (attack);
   v. maior defesa (defense);
   vi. mais rápido (speed);

In [1]:
```
# Importação das bibliotecas para a manipulação e visualização dos dados.

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from skimage.io import imread
%matplotlib inline
```

In [2]:
```
# Abertura do banco de dados e filtro das colunas que importam.

pokedata = pd.read_csv('pokedex_(Update_04.21).csv')
pokedata = pokedata.drop(columns = ['Unnamed: 0'])
pkcolumns = ['pokedex_number', 'name', 'german_name',
             'japanese_name', 'generation', 'status', 'species',
             'type_number', 'type_1', 'type_2', 'height_m',
             'weight_kg', 'hp', 'attack', 'defense',
             'sp_attack', 'sp_defense', 'speed', 'catch_rate']
pokedata = pokedata[pkcolumns]
pokedata.head(3)
```

Out[2]:

	pokedex_number	name	german_name	japanese_name	generation	status	species	type_number	type_1	type_2
0	1	Bulbasaur	Bisasam	フシギダネ (Fushigidane)	1	Normal	Seed Pokémon	2	Grass	Poison
1	2	Ivysaur	Bisaknosp	フシギソウ (Fushigisou)	1	Normal	Seed Pokémon	2	Grass	Poison
2	3	Venusaur	Bisaflor	フシギバナ (Fushigibana)	1	Normal	Seed Pokémon	2	Grass	Poison

1. Qual é a quantidade total de Pokémons neste dataset?

In [3]:
```
# Indica se há repetições no dataset

pokedata.duplicated().value_counts()
pkmcount = pokedata.name.nunique()

print(f'\nNeste dataset foram encontrados {pkmcount} pokémons (sem repetições).')
```

```
Neste dataset foram encontrados 1045 pokémons (sem repetições).
```

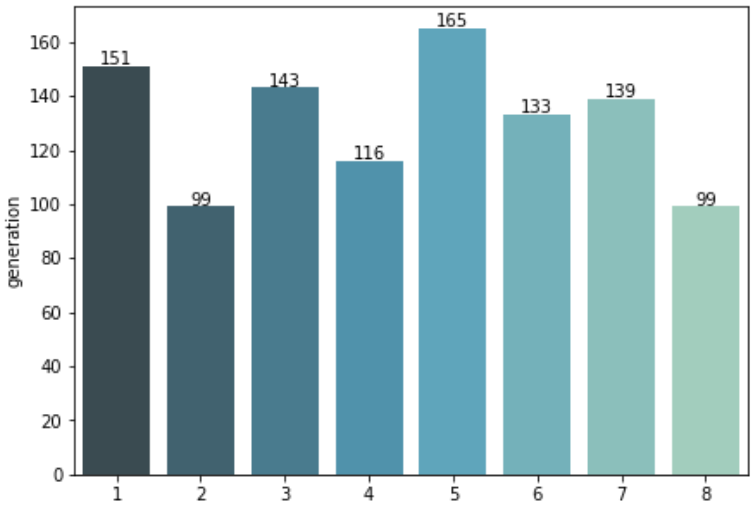
2. Quantos Pokemóns foram lancados em cada geracão?

Após avaliar o dataset, foi possível notar que alguns Pokemóns das primeiras gerações ganharam outras evoluções a partir das 6 e 7 gerações, porém constam como se fossem das primeiras. Dessa maneira as gerações desses Pokémons foram alteradas para as gerações corretas

In [4]:
```
# Atribuindo a geração correta aos PokémonsApós avaliar o dataset, foi possível notar que alguns Po
kemóns
# das primeiras gerações ganharam outras evoluções a partir das 6 e 7 gerações, porém constam como
 se fossem das primeiras.
# Dessa maneira as gerações desses Pokémons foram alteradas para as gerações corretas

pokedata.loc[pokedata['name'].str.contains('Mega'), 'generation'] = 6
pokedata.loc[pokedata['name'].str.contains('Alolan'), 'generation'] = 7
pokedata.loc[pokedata['name'].str.contains('Galarian'), 'generation'] = 7
pokedata.loc[pokedata['name'].str.contains('Partner'), 'generation'] = 7

# Contando a quantidade de Pokémons através da função 'value_counts'

pkbygen = pokedata.generation.value_counts().sort_index()

for i, j in enumerate(pkbygen):
    print(f'Na geração {i + 1} foram lançados {j} Pokémons')
```

```
Na geração 1 foram lançados 151 Pokémons
Na geração 2 foram lançados 99 Pokémons
Na geração 3 foram lançados 143 Pokémons
Na geração 4 foram lançados 116 Pokémons
Na geração 5 foram lançados 165 Pokémons
Na geração 6 foram lançados 133 Pokémons
Na geração 7 foram lançados 139 Pokémons
Na geração 8 foram lançados 99 Pokémons
```

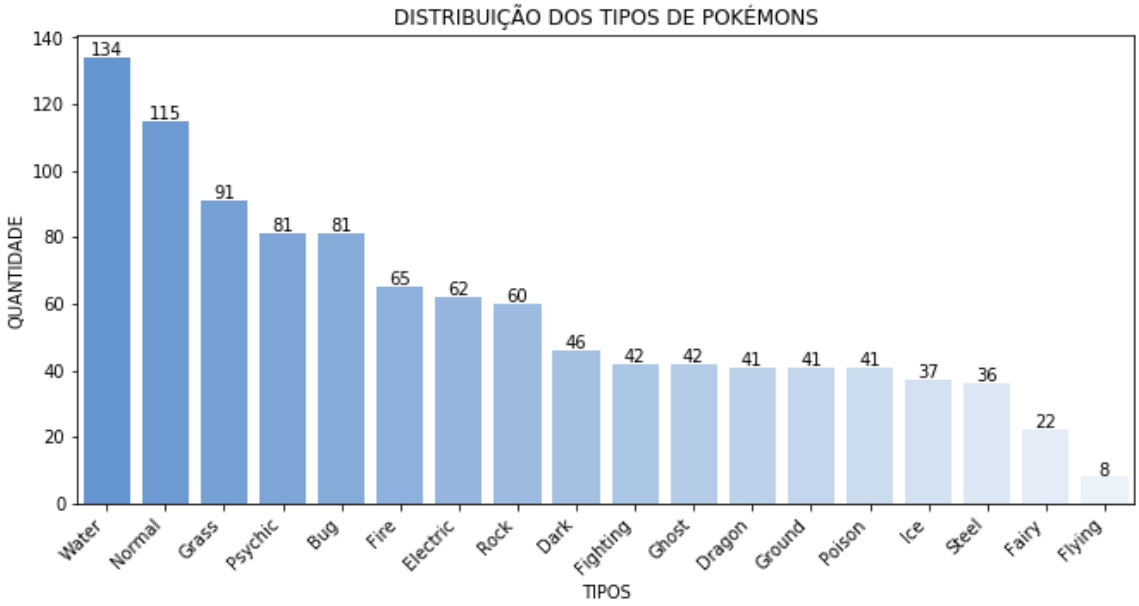
In [5]:
```
# Impressão de um barplot (gráfico de barras) de forma a representar a quantidade de pokémons por g
eração,
# neste caso, desde a 1ª até a 8ª geração

plt.figure(figsize=(7, 5))
pkg = sns.barplot(x = pkbygen.index, y = pkbygen, palette = 'GnBu_d')
for i in pkg.patches:
    pkg.annotate("%.0f" % i.get_height(), # retorna a altura das barras de acordo com as informaçõe
s nos patches
                 (i.get_x() + i.get_width() / 2, i.get_height()), # get_x = altura, get_y = largura
                  ha = 'center', # alinhamento do texto
                  va = 'baseline', # alinhamento vertical
                  color = 'black',
                  xytext = (0, 1),
                  textcoords = 'offset points',
                  fontsize = 10
                 )
```



3. Qual é a distribuição dos Pokémons por tipo_1 (predominante)?

In [6]:
```
# Separando os tipos de pokémons em um novo dataframe menor com as colunas 'type_1' e
# 'type_2'. Além disso, os valores 'NaN' foram substituídos por valores em branco ''.

types = pokedata[['type_1', 'type_2']]
types = types.fillna('')

# Contagem da distribuição dos Pokémons de acordo com o tipo predominante (type_1)
# Foram separados por duas variáveis x e y

x = types.type_1.value_counts().index.tolist()
y = types.type_1.value_counts()
```

In [7]:
```
# Plot da distribuição

fig, ax = plt.subplots(figsize = (11,5))
palette = sns.light_palette((250, 75, 60), n_colors = len(x), input="husl", reverse = True)

pkdist = sns.barplot(x=x, y=y, palette=palette)
for i in pkdist.patches:
    pkdist.annotate("%.0f" % i.get_height(), # retorna a altura das barras de acordo com as informações nos patches
                    (i.get_x() + i.get_width() / 2, i.get_height()), # get_x = altura, get_y = largura
                     ha = 'center', # alinhamento do texto
                     va = 'baseline', # alinhamento vertical
                     color = 'black', xytext = (0, 1), textcoords = 'offset points', fontsize = 10)
plt.title('DISTRIBUIÇÃO DOS TIPOS DE POKÉMONS', fontsize = 12)
plt.xticks(rotation = 45, horizontalalignment = 'right')
pkdist.set(xlabel = 'TIPOS', ylabel = 'QUANTIDADE')
plt.show()
```

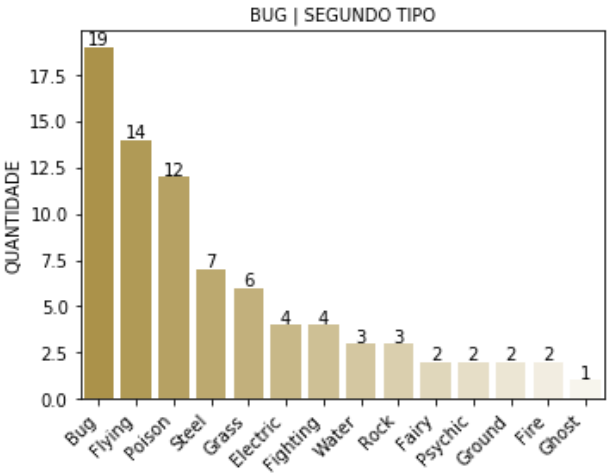


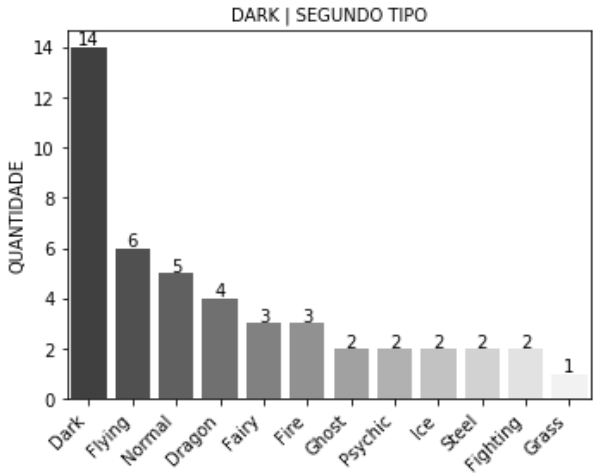
4. Qual é a distribuição dos Pokémons por tipo_2 em função do tipo_1?

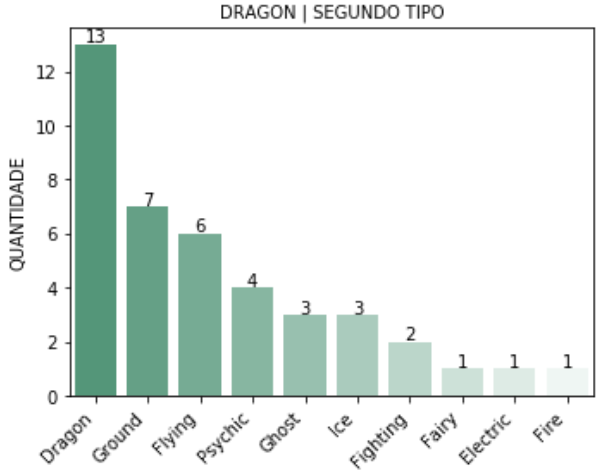
In [8]:

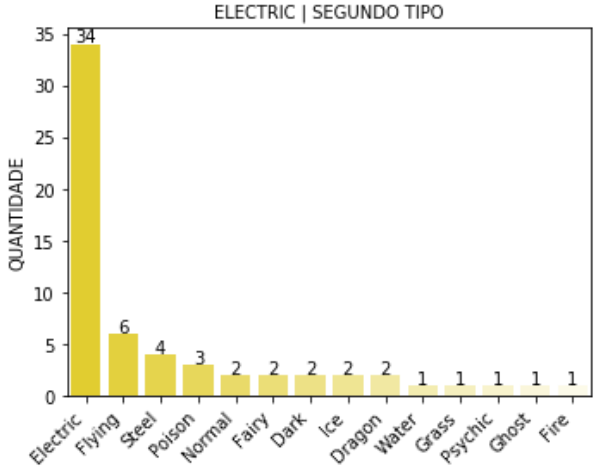
```
# Na coluna 'type_2', onde foram inseridos valores em branco (''), foram inseridos
# os valores predominantes. Como há Pokémons que não possuem mais de um tipo, foi
# mantido o tipo predominante como segundo tipo.

for i, j in enumerate(types.type_2):
    if j == '':
        types['type_2'][i] = types['type_1'][i]
    else:
        continue

# Construção de um dicionário para separação dos tipos secundários (type_2) e montagem dos gráficos

var = {}
pokevar = pokedata.type_1.value_counts().index.sort_values()
for i in pokevar:
    var[f'{i}'] = types[(types['type_1'] == i)]
```

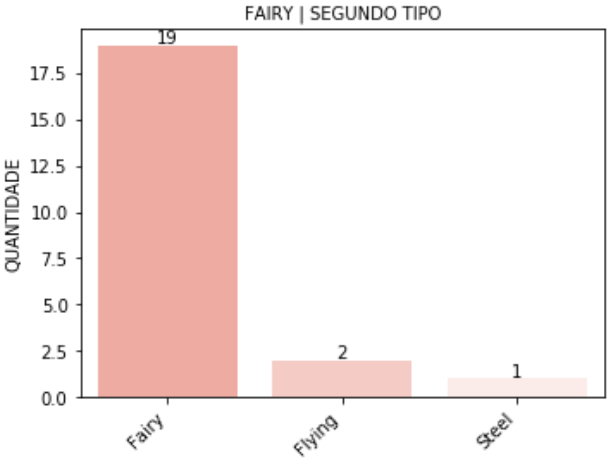
In [17]:
```
# Para deixar o gráfico visualmente mais bonito, separei algumas cores condizentes ao tipo de Pokém
on

dic = {'Bug': [65, 86, 65], 'Dark' : [63, 0, 27], 'Dragon': [150, 75, 60],
       'Electric' : [75, 99, 90], 'Fairy' : [20, 90, 75], 'Fighting' : [40, 86, 65],
       'Fire' : [13, 99, 50], 'Flying' : [250, 95, 80], 'Ghost' : [350, 95, 40],
       'Grass' : [130, 90, 65], 'Ground' : [63, 86, 65], 'Ice' : [250, 99, 60],
       'Normal' : [20, 90, 85], 'Poison' : [350, 95, 40], 'Psychic' : [50, 86, 60],
       'Rock' : [63, 0, 40], 'Steel' : [63, 0, 60], 'Water' : [250, 75, 60]}

colorspal = []

for i in dic.items():
    color = sns.light_palette((i[1]), n_colors = len(var[i[0]]['type_2'].value_counts()), input="hu
sl", reverse = True)
    colorspal.append(color)

# Geração de um loop for para coleta de todos as informações obtidas no dicionário. Nestes
# gráficos são demonstrados o tipo predominante e, em seguida, o tipo secundário dos pokémons

plt.figure(figsize = (12, 45))

for i, j in enumerate(pokevar):
    plt.subplot(9, 2, i + 1)
    g = sns.countplot('type_2', data = var[f'{j}'], order = var[f'{j}'].type_2.value_counts().index
, palette = colorspal[i])
    for i in g.patches:
        g.annotate("%.0f" % i.get_height(), # retorna a altura das barras de acordo com as informaç
ões nos patches
                   (i.get_x() + i.get_width() / 2, i.get_height()), # get_x = altura, get_y = lar
gura
                    ha = 'center', # alinhamento do texto
                    va = 'baseline', # alinhamento vertical
                    color = 'black', xytext = (0, 1), textcoords = 'offset points', fontsize = 10
)
    plt.title(f'{j.upper()} | SEGUNDO TIPO', fontsize = 10)
    g.set(xlabel = '', ylabel = 'QUANTIDADE')
    plt.xticks(rotation = 45, horizontalalignment = 'right', fontsize = 10)
plt.subplots_adjust(hspace=.3)
plt.show()
```

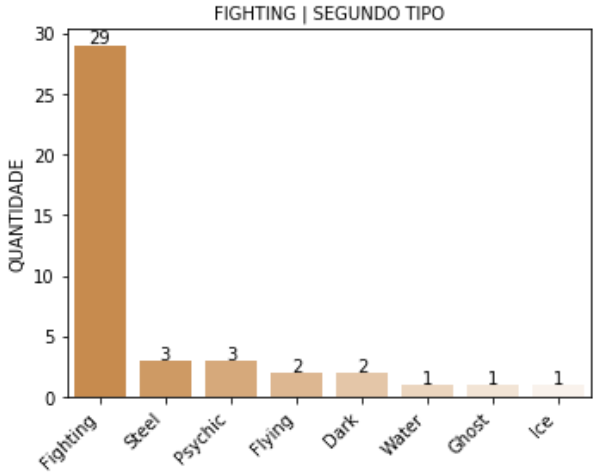

BUG | SEGUNDO TIPO
QUANTIDADE
Bug
Flying
Poison
Steel
Grass
Electric
Fighting
Water
Rock
Fairy
Psychic
Ground
Fire
Ghost

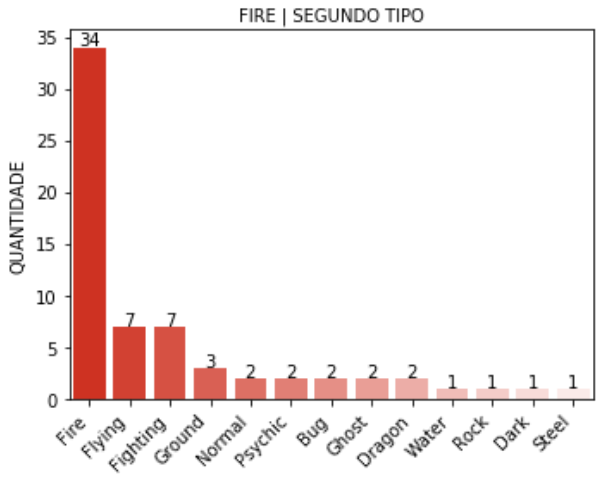

DARK | SEGUNDO TIPO
QUANTIDADE
Dark
Flying
Normal
Dragon
Fairy
Fire
Ghost
Psychic
Ice
Steel
Fighting
Grass

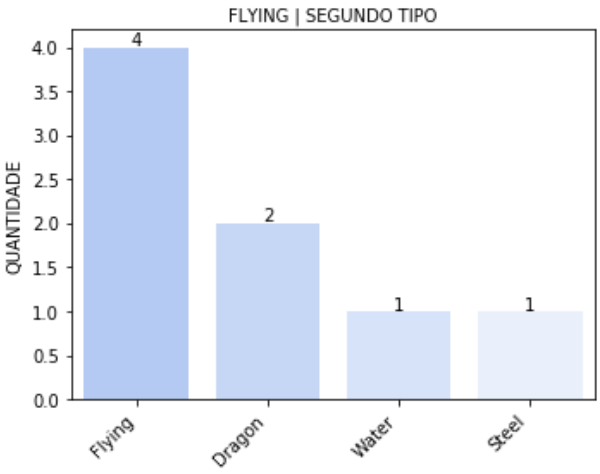

DRAGON | SEGUNDO TIPO
QUANTIDADE
Dragon
Ground
Flying
Psychic
Ghost
Ice
Fighting
Fairy
Electric
Fire

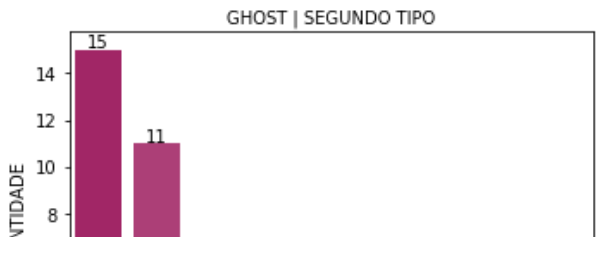

ELECTRIC | SEGUNDO TIPO
QUANTIDADE
Electric
Flying
Steel
Poison
Normal
Fairy
Dark
Ice
Dragon
Water
Grass
Psychic
Ghost
Fire

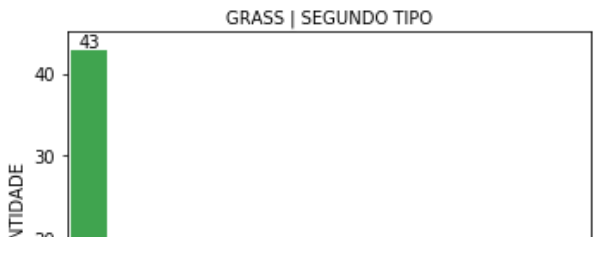

FAIRY | SEGUNDO TIPO
QUANTIDADE
Fairy
Flying
Steel

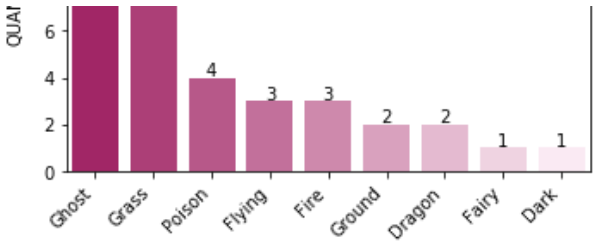

FIGHTING | SEGUNDO TIPO
QUANTIDADE
Fighting
Steel
Psychic
Flying
Dark
Water
Ghost
Ice

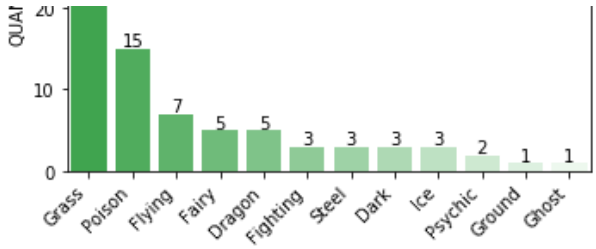

FIRE | SEGUNDO TIPO
QUANTIDADE
Fire
Flying
Fighting
Ground
Normal
Psychic
Bug
Ghost
Dragon
Water
Rock
Dark
Steel

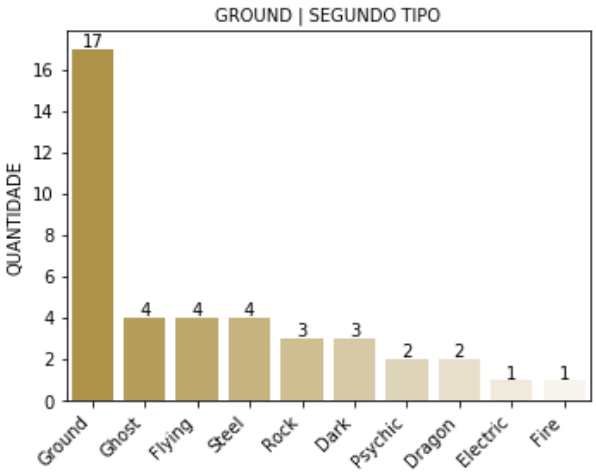

FLYING | SEGUNDO TIPO
QUANTIDADE
Flying
Dragon
Water
Steel

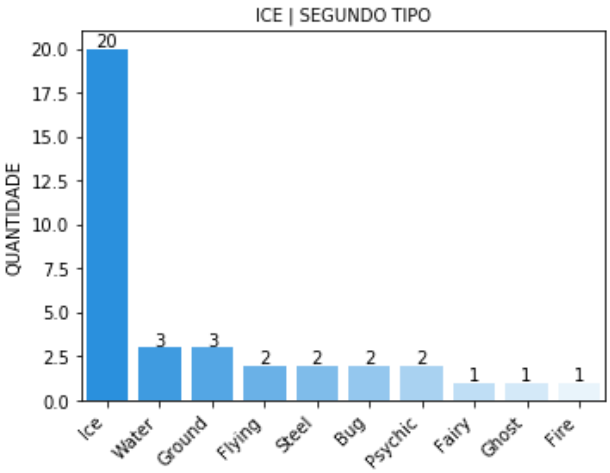

GHOST | SEGUNDO TIPO

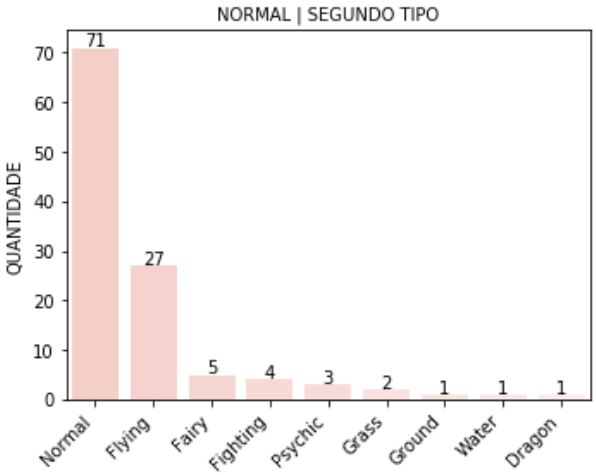

GRASS | SEGUNDO TIPO

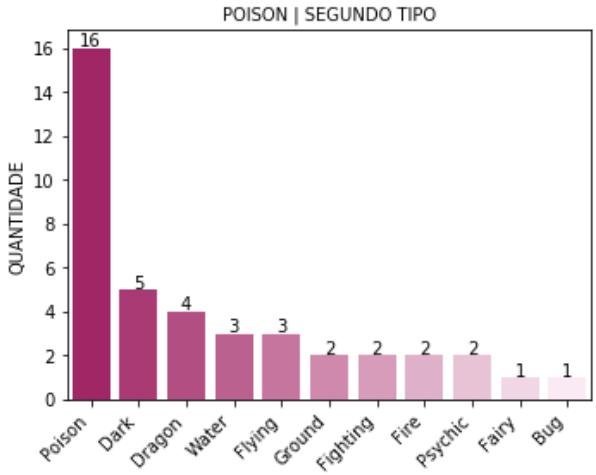

Ghost
Grass
Poison
Flying
Fire
Ground
Dragon
Fairy
Dark

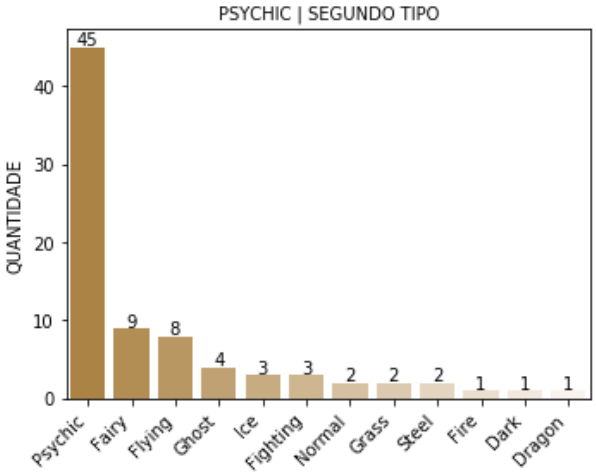

Grass
Poison
Flying
Fairy
Dragon
Fighting
Steel
Dark
Ice
Psychic
Ground
Ghost

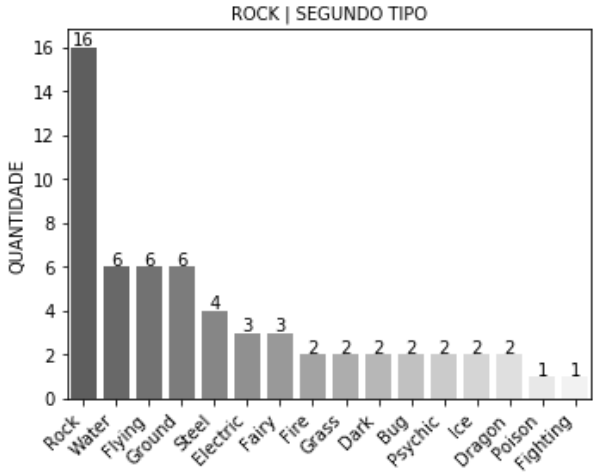

GROUND | SEGUNDO TIPO
QUANTIDADE
Ground
Ghost
Flying
Steel
Rock
Dark
Psychic
Dragon
Electric
Fire

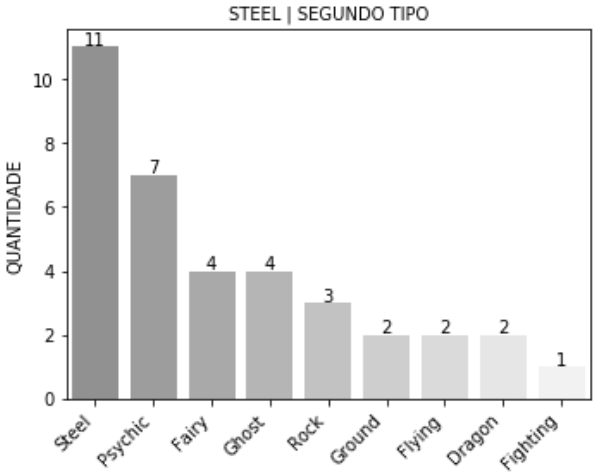

ICE | SEGUNDO TIPO
QUANTIDADE
Ice
Water
Ground
Flying
Steel
Bug
Psychic
Fairy
Ghost
Fire

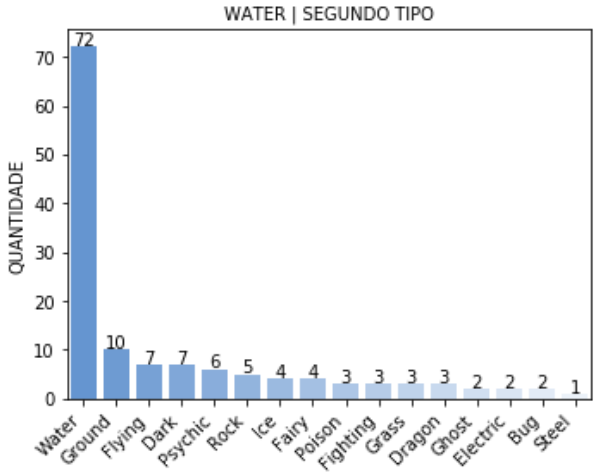

NORMAL | SEGUNDO TIPO
QUANTIDADE
Normal
Flying
Fairy
Fighting
Psychic
Grass
Ground
Water
Dragon


POISON | SEGUNDO TIPO
QUANTIDADE
Poison
Dark
Dragon
Water
Flying
Ground
Fighting
Fire
Psychic
Fairy
Bug


PSYCHIC | SEGUNDO TIPO
QUANTIDADE
Psychic
Fairy
Flying
Ghost
Ice
Fighting
Normal
Grass
Steel
Fire
Dark
Dragon


ROCK | SEGUNDO TIPO
QUANTIDADE
Rock
Water
Flying
Ground
Steel
Electric
Fairy
Fire
Grass
Dark
Bug
Psychic
Ice
Dragon
Poison
Fighting


STEEL | SEGUNDO TIPO
QUANTIDADE
Steel
Psychic
Fairy
Ghost
Rock
Ground
Flying
Dragon
Fighting


WATER | SEGUNDO TIPO
QUANTIDADE
Water
Ground
Flying
Dark
Psychic
Rock
Ice
Fairy
Poison
Fighting
Grass
Dragon
Ghost
Electric
Bug
Steel

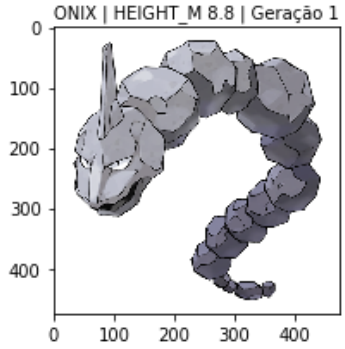
5. Em cada GERAÇÃO, identificar o Pokémon:

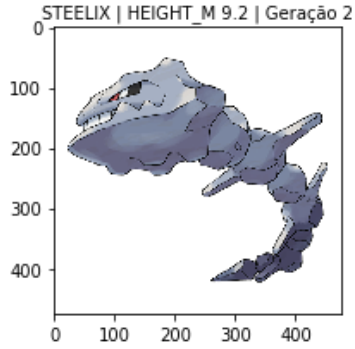
In [10]:
```
# Criação de uma função para coleta dos dados

def pokelytics(col = 'hp'):
    pdex = []
    pokelist = []
    pokenames = []

    print('='*70)
    print('{:^70}'.format(f'MAIORES VALORES POR GERAÇÃO: {col.upper()}'))
    print('='*70)
    print('|{:^16}|'.format('POKEDEX_NUMBER'), end = '')
    print('{:^25}|'.format('NOME'), end = '')
    print('{:^12}|'.format('GERAÇÃO'), end = '')
    print('{:^12}|'.format(col.upper()))

    # Busca e impressão das características de acordo com geração
    for i in range(1, 9, 1):
        hgen = pokedata[pokedata['generation'] == i][['pokedex_number','name', col]].sort_values(by
= [col], ascending = False)
        hgen_max = hgen.loc[hgen[col].idxmax()]
        pdex.append(hgen_max[0])
        pokelist.append(hgen_max.values[2])
        pokenames.append(hgen_max[1])

        print('|{:^16}|'.format(hgen_max.values[0]), end = '')
        print('{:^25}|'.format(hgen_max.values[1].upper()), end = '')
        print('{:^12}|'.format(i), end = '')
        print('{:^12}|'.format(hgen_max.values[2]))

    # Impressão das imagens dos Pokémons
    hp = []
    for i in pdex:
        if i == 103 or i == 150 or i == 65:
            hp.append(imread(str(f'https://assets.pokemon.com/assets/cms2/img/pokedex/full/{i:03d}_
f2.png')))
        else:
            hp.append(imread(str(f'https://assets.pokemon.com/assets/cms2/img/pokedex/full/{i:03d}.
png')))

    fig = plt.figure(figsize = (15, 15))

    for i in range(len(hp)):
        fig.add_subplot((len(hp)/2), 2, i + 1)
        plt.imshow(hp[i])
        name = pokedata[pokedata['name'] == pokenames[i]]['name'].values[0]
        gen = pokedata[pokedata['name'] == pokenames[i]]['generation'].values[0]
        plt.title(f'{pokenames[i].upper()} | {col.upper()} {pokelist[i]} | Geração {gen}', fontsize
= 10)
    plt.subplots_adjust(hspace=.3)
    plt.show()
```

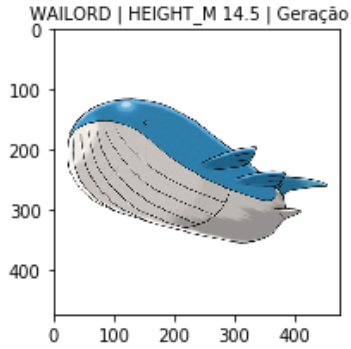
i. mais alto (height_m);

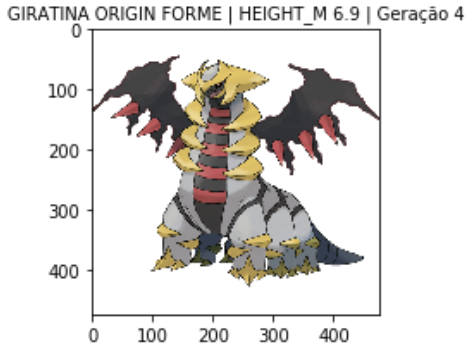
In [11]:
```
pokelytics(col = 'height_m')
```

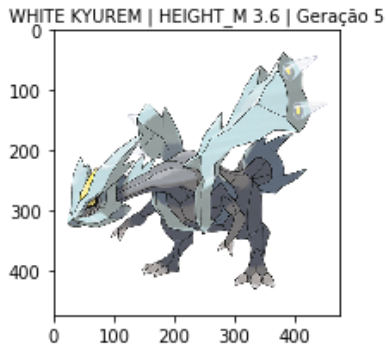
```
======================================================================
                MAIORES VALORES POR GERAÇÃO: HEIGHT_M
======================================================================
| POKEDEX_NUMBER |          NOME          |  GERAÇÃO  |  HEIGHT_M  |
|       95       |          ONIX          |     1     |     8.8    |
|      208       |        STEELIX         |     2     |     9.2    |
|      321       |        WAILORD         |     3     |     14.5   |
|      487       | GIRATINA ORIGIN FORME  |     4     |     6.9    |
|      646       |     WHITE KYUREM       |     5     |     3.6    |
|      384       |     MEGA RAYQUAZA      |     6     |     10.8   |
|      103       |   ALOLAN EXEGGUTOR     |     7     |     10.9   |
|      890       |  ETERNATUS ETERNAMAX   |     8     |    100.0   |
```

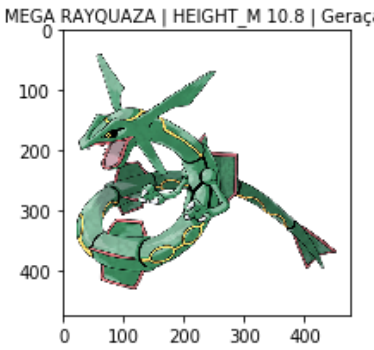


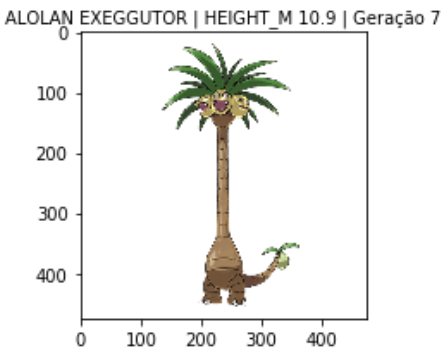


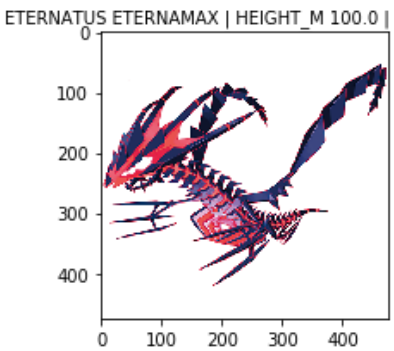








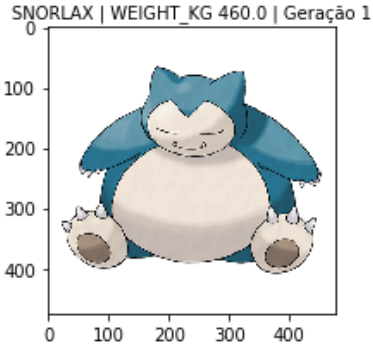


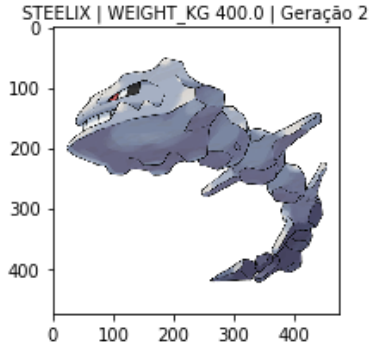


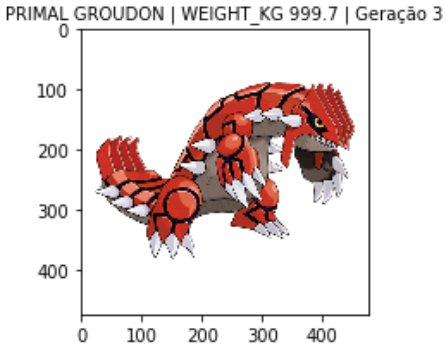
ii. mais pesado (weight_kg);

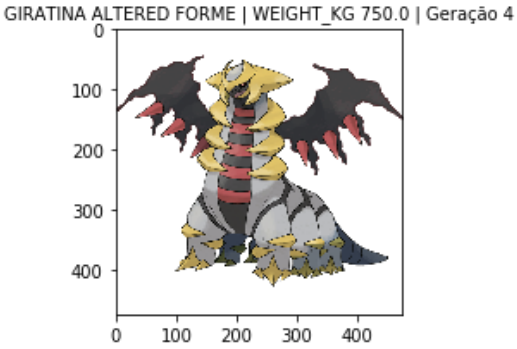
In [12]:
```
pokelytics(col = 'weight_kg')
```

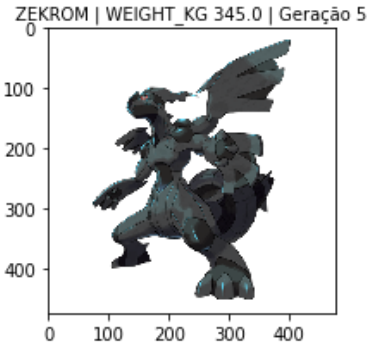
```
======================================================================
                MAIORES VALORES POR GERAÇÃO: WEIGHT_KG
======================================================================
| POKEDEX_NUMBER |          NOME          |  GERAÇÃO  | WEIGHT_KG  |
|      143       |         SNORLAX        |     1     |    460.0   |
|      208       |         STEELIX        |     2     |    400.0   |
|      383       |     PRIMAL GROUDON     |     3     |    999.7   |
|      487       | GIRATINA ALTERED FORME |     4     |    750.0   |
|      644       |         ZEKROM         |     5     |    345.0   |
|      376       |     MEGA METAGROSS     |     6     |    942.9   |
|      790       |         COSMOEM        |     7     |    999.9   |
|      890       |        ETERNATUS       |     8     |    950.0   |
```

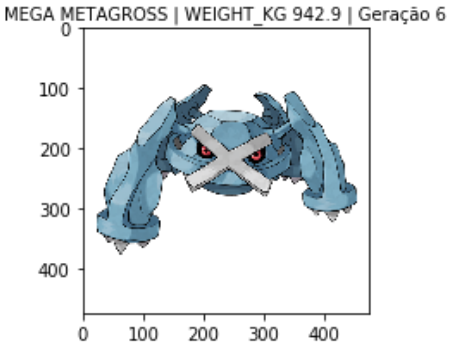


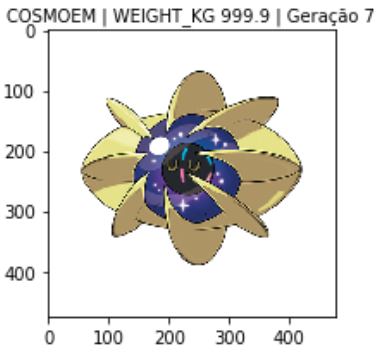


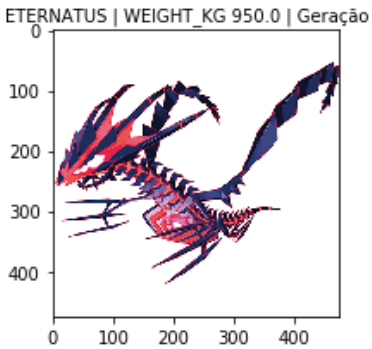








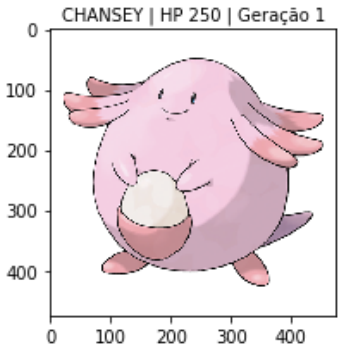


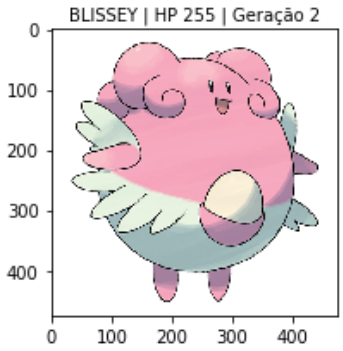


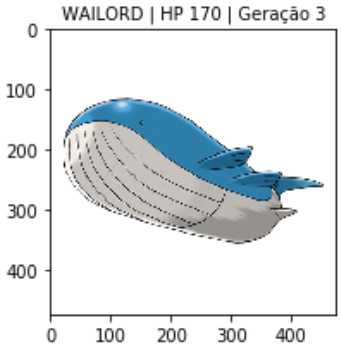
iii. maior HP;

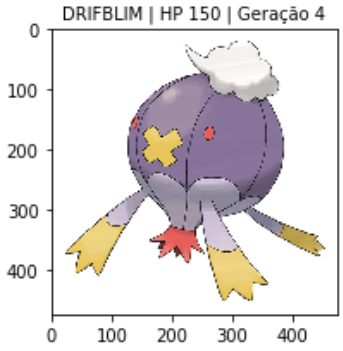
In [13]:
```
pokelytics(col = 'hp')
```

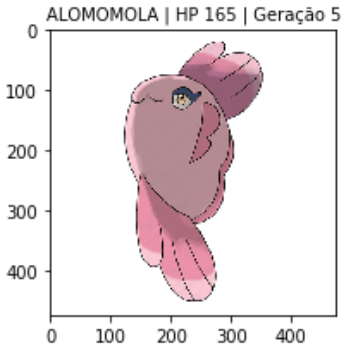
```
======================================================================
                    MAIORES VALORES POR GERAÇÃO: HP
======================================================================
| POKEDEX_NUMBER |          NOME          | GERAÇÃO |    HP    |
|      113       |         CHANSEY        |    1    |    250   |
|      242       |         BLISSEY        |    2    |    255   |
|      321       |         WAILORD        |    3    |    170   |
|      426       |        DRIFBLIM        |    4    |    150   |
|      594       |        ALOMOMOLA       |    5    |    165   |
|      718       | ZYGARDE COMPLETE FORME |    6    |    216   |
|      799       |        GUZZLORD        |    7    |    223   |
|      890       |   ETERNATUS ETERNAMAX  |    8    |    255   |
```

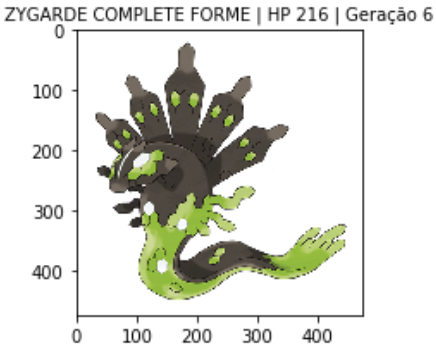


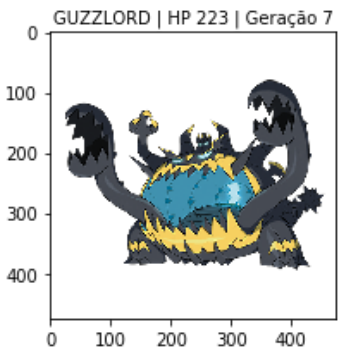


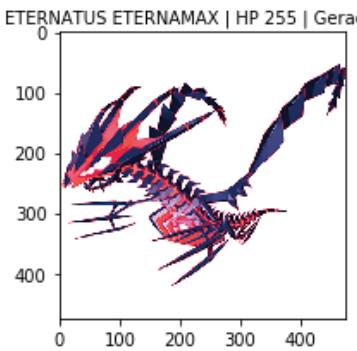








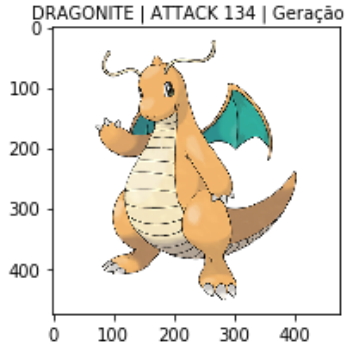


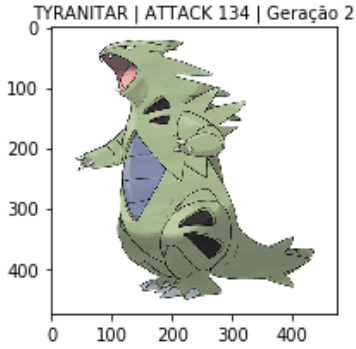


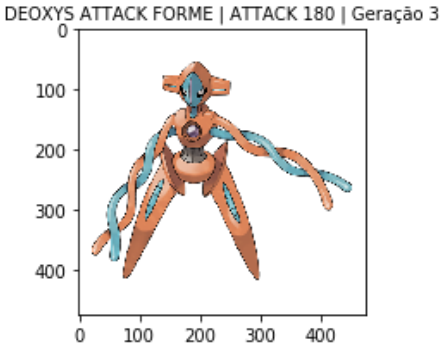
iv. maior ataque (attack);

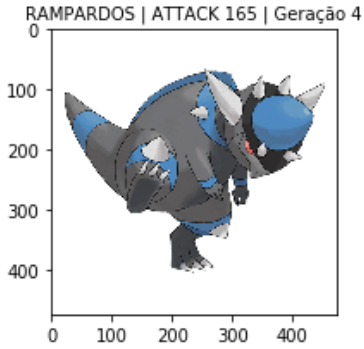
In [14]:
```
pokelytics(col = 'attack')
```

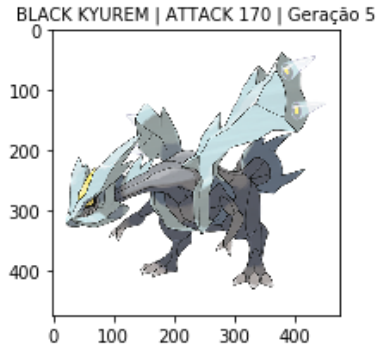
```
======================================================================
                MAIORES VALORES POR GERAÇÃO: ATTACK
======================================================================
| POKEDEX_NUMBER |          NOME          | GERAÇÃO |  ATTACK  |
|      149       |       DRAGONITE        |    1    |   134    |
|      248       |       TYRANITAR        |    2    |   134    |
|      386       |  DEOXYS ATTACK FORME   |    3    |   180    |
|      409       |       RAMPARDOS        |    4    |   165    |
|      646       |      BLACK KYUREM      |    5    |   170    |
|      150       |      MEGA MEWTWO X     |    6    |   190    |
|      798       |        KARTANA         |    7    |   181    |
|      888       | ZACIAN CROWNED SWORD   |    8    |   170    |
```

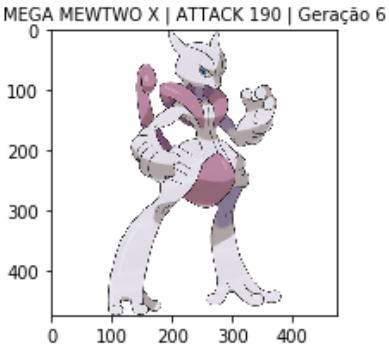


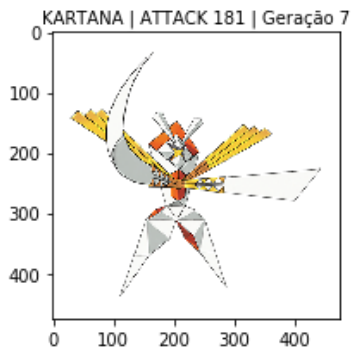


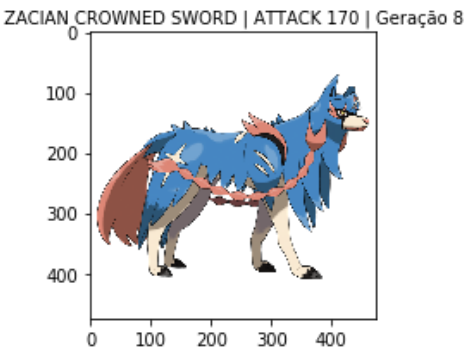








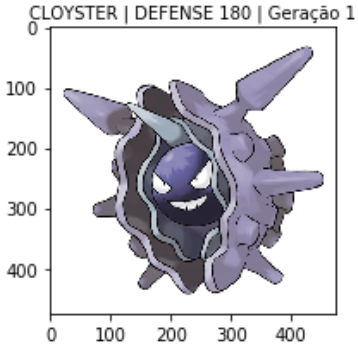


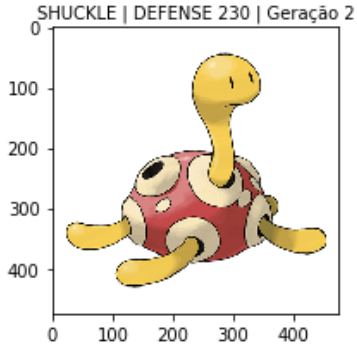


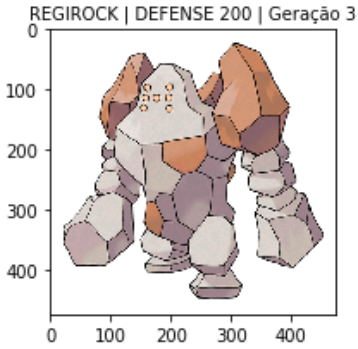
v. maior defesa;

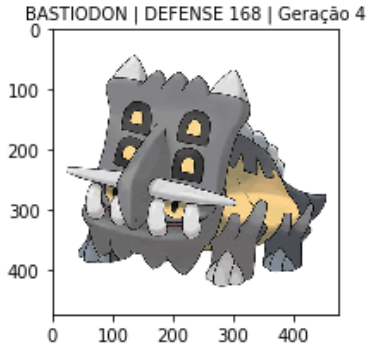
In [15]:
```
pokelytics(col = 'defense')
```

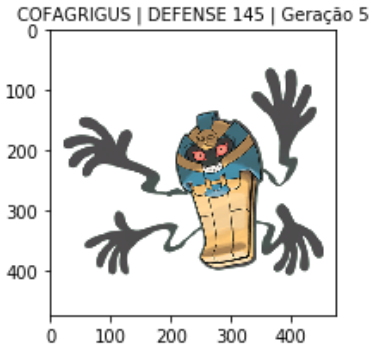
```
======================================================================
                MAIORES VALORES POR GERAÇÃO: DEFENSE
======================================================================
| POKEDEX_NUMBER |          NOME          | GERAÇÃO |  DEFENSE  |
|       91       |        CLOYSTER        |    1    |    180    |
|      213       |         SHUCKLE        |    2    |    230    |
|      377       |        REGIROCK        |    3    |    200    |
|      411       |       BASTIODON        |    4    |    168    |
|      563       |      COFAGRIGUS        |    5    |    145    |
|      306       |      MEGA AGGRON       |    6    |    230    |
|      805       |       STAKATAKA        |    7    |    211    |
|      890       |  ETERNATUS ETERNAMAX   |    8    |    250    |
```

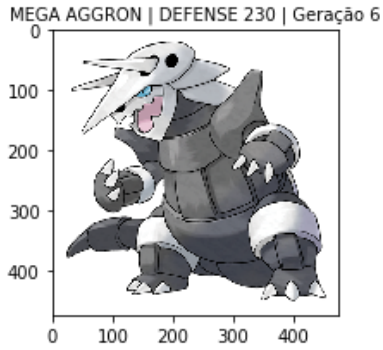


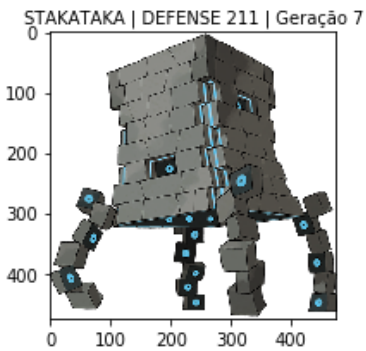


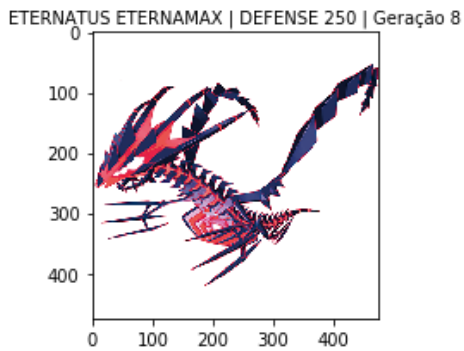












vi. mais rápido;

In [16]:
```
pokelytics(col = 'speed')
```

```
======================================================================
                    MAIORES VALORES POR GERAÇÃO: SPEED
======================================================================
| POKEDEX_NUMBER |          NOME          | GERAÇÃO |  SPEED  |
|      101       |       ELECTRODE        |    1    |   150   |
|      169       |        CROBAT          |    2    |   130   |
|      386       |  DEOXYS SPEED FORME    |    3    |   180   |
|      492       |   SHAYMIN SKY FORME    |    4    |   127   |
|      617       |       ACCELGOR         |    5    |   145   |
|       65       |     MEGA ALAKAZAM      |    6    |   150   |
|      795       |       PHEROMOSA        |    7    |   151   |
|      894       |       REGIELEKI        |    8    |   200   |
```