Pokemon Trivia

In this module we will use pandas module to answer various questions related to pokemons by moulding and twisting their database accordingly.

In [2]:
```
#getting the dataset
import pandas as pd
pokemon=pd.read_csv("C:\\Users\\sujoydutta\\Downloads\\pokemon.csv")
pokemon.head()
```

Out[2]:

	abilities	against_bug	against_dark	against_dragon	against_electric	against_fairy	against_fight	against_fi
0	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2
1	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2
2	['Overgrow', 'Chlorophyll']	1.0	1.0	1.0	0.5	0.5	0.5	2
3	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	(
4	['Blaze', 'Solar Power']	0.5	1.0	1.0	1.0	0.5	1.0	(

5 rows × 41 columns

In [3]:
```
#examining the dataset
pokemon.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 801 entries, 0 to 800
Data columns (total 41 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   abilities          801 non-null    object 
 1   against_bug        801 non-null    float64
 2   against_dark       801 non-null    float64
 3   against_dragon     801 non-null    float64
 4   against_electric   801 non-null    float64
 5   against_fairy      801 non-null    float64
 6   against_fight      801 non-null    float64
 7   against_fire       801 non-null    float64
 8   against_flying     801 non-null    float64
 9   against_ghost      801 non-null    float64
 10  against_grass      801 non-null    float64
 11  against_ground     801 non-null    float64
 12  against_ice        801 non-null    float64
 13  against_normal     801 non-null    float64
 14  against_poison     801 non-null    float64
 15  against_psychic    801 non-null    float64
 16  against_rock       801 non-null    float64
 17  against_steel      801 non-null    float64
 18  against_water      801 non-null    float64
 19  attack             801 non-null    int64  
 20  base_egg_steps     801 non-null    int64  
 21  base_happiness     801 non-null    int64  
```

```
 22  base_total          801 non-null    int64
 23  capture_rate        801 non-null    object
 24  classfication       801 non-null    object
 25  defense             801 non-null    int64
 26  experience_growth   801 non-null    int64
 27  height_m            781 non-null    float64
 28  hp                  801 non-null    int64
 29  japanese_name       801 non-null    object
 30  name                801 non-null    object
 31  percentage_male     703 non-null    float64
 32  pokedex_number      801 non-null    int64
 33  sp_attack           801 non-null    int64
 34  sp_defense          801 non-null    int64
 35  speed               801 non-null    int64
 36  type1               801 non-null    object
 37  type2               417 non-null    object
 38  weight_kg           781 non-null    float64
 39  generation          801 non-null    int64
 40  is_legendary        801 non-null    int64
dtypes: float64(21), int64(13), object(7)
memory usage: 256.7+ KB
```

In [4]:
```
#counting pokemons in first seven generations
pokemon['generation'].count()
```

Out[4]:
```
801
```

In [5]:
```
#seeing total columns
total_columns = len(pokemon.columns)
print(total_columns)
```

```
41
```

In [6]:
```
#genderless pokemons

genderless = pokemon['percentage_male'].isnull().sum()
print(genderless)
```

```
98
```

In [7]:
```
#female pokemon
pokemon['percentage_female']= 100-pokemon['percentage_male']
pokemon['percentage_female']
```

Out[7]:
```
0      11.9
1      11.9
2      11.9
3      11.9
4      11.9
       ...
796     NaN
797     NaN
798     NaN
799     NaN
800     NaN
Name: percentage_female, Length: 801, dtype: float64
```

In [8]:
```
#seeing which species has more female
pokemonfemale = pokemon[['name', 'percentage_female']]
pokemonfemale_sorted = pokemonfemale.sort_values(by='percentage_female', ascending=False
pokemonfemale_sorted.head(10)
```

Out[8]:

	name	percentage_female
487	Cresselia	100.0

547	Petilil	100.0
760	Bounsweet	100.0
761	Steenee	100.0
762	Tsareena	100.0
439	Happiny	100.0
379	Latias	100.0
30	Nidoqueen	100.0
29	Nidorina	100.0
28	Nidoran♀	100.0

In [9]:
```
#pokemon types
pokemon.type1.value_counts()
```

Out[9]:
```
type1
water       114
normal      105
grass        78
bug          72
psychic      53
fire         52
rock         45
electric     39
poison       32
ground       32
dark         29
fighting     28
ghost        27
dragon       27
steel        24
ice          23
fairy        18
flying        3
Name: count, dtype: int64
```

In [10]:
```
#dual type pokemon counts
dualpokemon= pokemon[['type1', 'type2']]
dualpokemon=dualpokemon.dropna()
len(dualpokemon)
```

Out[10]:
```
417
```

In [21]:
```
#Top 10 strongest pokemons
pokemonstrong = pokemon[['name', 'base_total']]
pokemonstrong_sorted = pokemonstrong.sort_values(by='base_total', ascending=False)
top_10_strongest_pokemon = pokemonstrong_sorted.head(10)
print(top_10_strongest_pokemon)
```

```
          name  base_total
149     Mewtwo         780
383   Rayquaza         780
382    Groudon         770
381     Kyogre         770
492     Arceus         720
717    Zygarde         708
380     Latios         700
372  Salamence         700
247  Tyranitar         700
444   Garchomp         700
```

In [25]:
```
# Filter for non-legendary Pokémon
nonlegendarystrong = pokemon[['name', 'base_total', 'is_legendary']]
nonlegendarystrong = nonlegendarystrong[nonlegendarystrong['is_legendary'] == 0]


nonlegendarystrong_sorted = nonlegendarystrong.sort_values(by='base_total', ascending=Fa

top_5_strongest_pokemon = nonlegendarystrong_sorted.head(5)

print(top_5_strongest_pokemon)
```

```
          name  base_total  is_legendary
372  Salamence         700             0
375  Metagross         700             0
444   Garchomp         700             0
247  Tyranitar         700             0
288    Slaking         670             0
```

In [26]:
```
# Top 10 lighest pokemon
lowweightpokemon = pokemon[['name','weight_kg']]

lowweightpokemon_sorted = lowweightpokemon.sort_values(by='weight_kg', ascending=True)

top_10_lightest_pokemon = lowweightpokemon_sorted .head(10)

print(top_10_lightest_pokemon)
```

```
           name  weight_kg
91       Gastly        0.1
788      Cosmog        0.1
797     Kartana        0.1
92      Haunter        0.1
668     Flabébé        0.1
741    Cutiefly        0.2
745  Wishiwashi        0.3
763      Comfey        0.3
601      Tynamo        0.3
478       Rotom        0.3
```

In [27]:
```
#descriptive statisics for weight
pokemon['weight_kg'].describe()
```

Out[27]:
```
count    781.000000
mean      61.378105
std      109.354766
min        0.100000
25%        9.000000
50%       27.300000
75%       64.800000
max      999.900000
Name: weight_kg, dtype: float64
```

In [29]:
```
#Seeing pokemon counts by generation
pokemon['generation'].value_counts()
```

Out[29]:
```
generation
5    156
1    151
3    135
4    107
2    100
7     80
6     72
Name: count, dtype: int64
```

In [30]:
```
#Pokemon and abilities
pokemonability=pokemon[['name', 'abilities']]
```

```
pokemonability.head(10)
```

Out[30]:

	name	abilities
0	Bulbasaur	['Overgrow', 'Chlorophyll']
1	Ivysaur	['Overgrow', 'Chlorophyll']
2	Venusaur	['Overgrow', 'Chlorophyll']
3	Charmander	['Blaze', 'Solar Power']
4	Charmeleon	['Blaze', 'Solar Power']
5	Charizard	['Blaze', 'Solar Power']
6	Squirtle	['Torrent', 'Rain Dish']
7	Wartortle	['Torrent', 'Rain Dish']
8	Blastoise	['Torrent', 'Rain Dish']
9	Caterpie	['Shield Dust', 'Run Away']

In [34]:

```
# Selecting the fastest pokemon by average speed

average_speed = pokemon.groupby('name')['speed'].mean().reset_index()

average_speed.columns = ['name', 'average_speed']

average_speed_sorted = average_speed.sort_values(by='average_speed', ascending=False)

average_speed_sorted.head(1)
```

Out[34]:

	name	average_speed
152	Deoxys	180.0

In [42]:

```
#pokemon type
pokemon['type']= pokemon.apply(lambda row: 'only'+' '+row['type1'] if pd.isna(row['type2
)

pokemon['type'].head()
```

Out[42]:

```
0    grass and poison
1    grass and poison
2    grass and poison
3           only fire
4           only fire
Name: type, dtype: object
```

In [44]:

```
#  pokemon types that deal double damage against fire type pokemons
doubledmgagainstfire = pokemon[['type','against_fire']]
doubledmgagainstfire = doubledmgagainstfire[doubledmgagainstfire['against_fire'] == 2]
doubledmgagainstfire['type'].unique()
```

Out[44]:

```
array(['grass and poison', 'only bug', 'bug and flying', 'bug and poison',
       'electric and steel', 'grass and psychic', 'only grass',
       'ice and psychic', 'ice and flying', 'grass and flying',
       'steel and ground', 'bug and fighting', 'dark and ice',
       'ice and ground', 'steel and flying', 'psychic and grass',
       'grass and dark', 'grass and fighting', 'bug and ground',
       'bug and ghost', 'steel and fairy', 'only ice',
       'steel and psychic', 'only steel', 'grass and ground',
       'fighting and steel', 'poison and bug', 'ice and ghost',
       'grass and grass', 'ground and steel', 'grass and fairy',
```

```
       'normal and grass', 'bug and electric', 'dark and steel',
       'steel and fighting', 'steel and ghost', 'ghost and grass',
       'grass and ghost', 'fighting and ice', 'bug and fairy',
       'psychic and steel'], dtype=object)
```

In [45]:
```
# types that are strong against rock type pokemons
strongagainstrock = pokemon[['type','against_rock']]
strongagainstrock = strongagainstrock[strongagainstrock['against_rock'] >1]
strongagainstrock['type'].unique()
```

Out[45]:
```
array(['only fire', 'fire and flying', 'only bug', 'bug and flying',
       'bug and poison', 'normal and flying', 'fire and ice',
       'poison and flying', 'bug and grass', 'water and ice',
       'ice and psychic', 'water and flying', 'rock and flying',
       'ice and flying', 'electric and flying', 'dragon and flying',
       'fairy and flying', 'psychic and flying', 'grass and flying',
       'dark and flying', 'bug and rock', 'dark and ice', 'fire and rock',
       'dark and fire', 'bug and water', 'bug and ghost', 'rock and bug',
       'only ice', 'ice and water', 'ghost and flying', 'poison and bug',
       'grass and ice', 'ice and ghost', 'psychic and fire',
       'fire and fire', 'bug and electric', 'ghost and fire',
       'bug and fire', 'only flying', 'dragon and fire', 'dragon and ice',
       'fire and psychic', 'fire and normal', 'rock and ice',
       'flying and dragon', 'fire and water', 'fire and dark',
       'bug and fairy', 'water and bug', 'poison and fire',
       'fire and dragon'], dtype=object)
```

In [48]:
```
#pokemon bmi analysis
pokemon['bmi']=pokemon['weight_kg']/pokemon['height_m']**2
pokemonbmi=pokemon[['name','bmi']]
bmisorted=pokemonbmi.sort_values(by='bmi', ascending=False)

bmisorted.head(10)
```

Out[48]:

	name	bmi
789	Cosmoem	99990.000000
773	Minior	444.444444
303	Aron	375.000000
631	Durant	366.666667
365	Clamperl	328.125000
323	Torkoal	321.600000
330	Cacnea	320.625000
445	Munchlax	291.666667
768	Sandygast	280.000000
373	Beldum	264.444444

In [49]:
```
pokemon.columns
```

Out[49]:
```
Index(['abilities', 'against_bug', 'against_dark', 'against_dragon',
       'against_electric', 'against_fairy', 'against_fight', 'against_fire',
       'against_flying', 'against_ghost', 'against_grass', 'against_ground',
       'against_ice', 'against_normal', 'against_poison', 'against_psychic',
       'against_rock', 'against_steel', 'against_water', 'attack',
       'base_egg_steps', 'base_happiness', 'base_total', 'capture_rate',
       'classfication', 'defense', 'experience_growth', 'height_m', 'hp',
       'japanese_name', 'name', 'percentage_male', 'pokedex_number',
       'sp_attack', 'sp_defense', 'speed', 'type1', 'type2', 'weight_kg',
```

```
       'generation', 'is_legendary', 'percentage_female', 'type', 'bmi'],
      dtype='object')
```

In [59]:

```
#seeing which combination has the fewest weaknesses

damage_columns = [col for col in pokemon.columns if col.startswith('against_')]


type_damage = pokemon.groupby('type')[damage_columns].min().reset_index()


type_damage['min_damage'] = type_damage[damage_columns].min(axis=1)

strongest_type = type_damage.loc[type_damage['min_damage'].idxmax()]

print(strongest_type)
```

```
type                bug and electric
against_bug                      1.0
against_dark                     1.0
against_dragon                   1.0
against_electric                 0.5
against_fairy                    1.0
against_fight                    0.5
against_fire                     2.0
against_flying                   1.0
against_ghost                    1.0
against_grass                    0.5
against_ground                   1.0
against_ice                      1.0
against_normal                   1.0
against_poison                   1.0
against_psychic                  1.0
against_rock                     2.0
against_steel                    0.5
against_water                    1.0
min_damage                       0.5
Name: 0, dtype: object
```