Ejercicio 5

Inicializa una sesión de spark en consola y coloca en una variable la información de cada archivo csv que se proporciona:

- Realiza una unión para unir todos los pokemon de cada generación
- Devuelve un conteo por generación y muestra el resultado en pantalla
- Devuelve un conteo por Type_1 y muestra el resultado en pantalla
- Devuelve un conteo por pokemon legendarios y muestra el resultado en pantalla
- Devuelve el pokemon con más puntos de:
  - Vida (hp)
  - Ataque (attack)
  - Defensa (defense)
- Devuelve el pokemon con menos puntos de:
  - Ataque especial (sp_attack)
  - Defensa especial (sp_defense)
  - Velocidad (speed)
- Realiza un Join de la información de los pokemon y sus ventajas
- Del resultado final, devuelve toda la información del pokemon, pero solo la información de las primeras 2 ventajas, las primeras 2 desventajas y el campo “sin_efecto”
- Escribe el resultado en un archivo parquet pkmn_result__info.parquet

Importaciones y creacion se SparkSession y SparkContext

In [2]:
```
import findspark
findspark.init()

import pandas as pd
import pyspark
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
sc = spark.sparkContext
```

Punto #1

coloca en una variable la información de cada archivo csv que se proporciona:

In [5]:
```
"""
Usaremos esta funcion de forma auxiliar para darle formato a los dataframes ya que
se infiere que los numeros son String y arreglaremos este problema para evitar posible
errores al manipular los valores
"""
```

```
from pyspark.sql.types import FloatType
from pyspark.sql.functions import regexp_replace


def formatPkmnDataframe(path):
    dataframe = spark.read.csv(path, header=True, inferSchema=True)
    columnas = ["hp", "attack", "defense", "sp_atk", "sp_def", "speed"]
    for columna in columnas:
        dataframe = dataframe.withColumn(columna, regexp_replace(columna, ",", "."))
        dataframe = dataframe.withColumn(columna, dataframe[columna].cast(FloatType()))
    return dataframe
```

In [6]:
```
pkmn_gen_1 = formatPkmnDataframe("Data/pkmn_gen_1.csv")
pkmn_gen_2 = formatPkmnDataframe("Data/pkmn_gen_2.csv")
pkmn_gen_3 = formatPkmnDataframe("Data/pkmn_gen_3.csv")
pkmn_gen_4 = formatPkmnDataframe("Data/pkmn_gen_4.csv")
pkmn_gen_5 = formatPkmnDataframe("Data/pkmn_gen_5.csv")
pkmn_gen_6 = formatPkmnDataframe("Data/pkmn_gen_6.csv")
```

In [7]:
```
pkmn_gen_1.printSchema()
```

```
root
 |-- id: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- type_1: string (nullable = true)
 |-- type_2: string (nullable = true)
 |-- hp: float (nullable = true)
 |-- attack: float (nullable = true)
 |-- defense: float (nullable = true)
 |-- sp_atk: float (nullable = true)
 |-- sp_def: float (nullable = true)
 |-- speed: float (nullable = true)
 |-- legendary: boolean (nullable = true)
```

Punto #2

Realiza una unión para unir todos los pokemon de cada generación

In [9]:
```
from functools import reduce
from pyspark.sql import DataFrame

all_pkmn = [pkmn_gen_1, pkmn_gen_2, pkmn_gen_3, pkmn_gen_4, pkmn_gen_5, pkmn_gen_6]

# Utilizamos reduce para unir todos los DataFrames en uno solo
all_pkmn = reduce(DataFrame.union, all_pkmn)

all_pkmn.show(5)
```

```
+---+--------------------+------+------+----+------+-------+------+------+-----+-----
----+
| id|                name|type_1|type_2|  hp|attack|defense|sp_atk|sp_def|speed|legen
dary|
+---+--------------------+------+------+----+------+-------+------+------+-----+-----
----+
|  1|           Bulbasaur| Grass|Poison|45.0|  49.0|   49.0|  65.0|  65.0| 45.0|    f
alse|
|  2|             Ivysaur| Grass|Poison|60.0|  62.0|   63.0|  80.0|  80.0| 60.0|    f
alse|
|  3|            Venusaur| Grass|Poison|80.0|  82.0|   83.0| 100.0| 100.0| 80.0|    f
alse|
|  3|VenusaurMega Venu...| Grass|Poison|80.0| 100.0|  123.0| 122.0| 120.0| 80.0|    f
alse|
|  4|          Charmander|  Fire|  null|39.0|  52.0|   43.0|  60.0|  50.0| 65.0|    f
alse|
+---+--------------------+------+------+----+------+-------+------+------+-----+-----
----+
only showing top 5 rows
```

Punto #3

Devuelve un conteo por generación y muestra el resultado en pantalla

In [11]:
```
print("Existen {} pokemon en la 1° generacion".format(pkmn_gen_1.count()))
print("Existen {} pokemon en la 2° generacion".format(pkmn_gen_2.count()))
print("Existen {} pokemon en la 3° generacion".format(pkmn_gen_3.count()))
print("Existen {} pokemon en la 4° generacion".format(pkmn_gen_4.count()))
print("Existen {} pokemon en la 5° generacion".format(pkmn_gen_5.count()))
print("Existen {} pokemon en la 6° generacion".format(pkmn_gen_6.count()))
```

```
Existen 166 pokemon en la 1° generacion
Existen 106 pokemon en la 2° generacion
Existen 160 pokemon en la 3° generacion
Existen 121 pokemon en la 4° generacion
Existen 165 pokemon en la 5° generacion
Existen 82 pokemon en la 6° generacion
```

Punto #4

Devuelve un conteo por Type_1 y muestra el resultado en pantalla

In [19]:
```
from pyspark.sql.functions import count
print("Cantidad de pokemon por tipo de la generacion 1 a la 6")
pkmn_type1 = all_pkmn.groupBy("type_1").agg(count("type_1").alias("Cantidad")).show()
```

```
Cantidad de pokemon por tipo de la generacion 1 a la 6
+--------+--------+
|  type_1|Cantidad|
+--------+--------+
|   Water|     112|
|  Poison|      28|
|    Rock|      44|
|     Ice|      24|
|   Ghost|      32|
| Psychic|      57|
|  Dragon|      32|
|     Bug|      69|
|Electric|      44|
|    Fire|      52|
|  Ground|      32|
|Fighting|      27|
|   Grass|      70|
|  Normal|     100|
|   Steel|      27|
|   Fairy|      15|
|    Dark|      31|
|  Flying|       4|
+--------+--------+
```

Punto #5

Devuelve un conteo por pokemon legendarios y muestra el resultado en pantalla

In [28]:
```
cantidad = all_pkmn.filter(all_pkmn["legendary"] == True).count()
print("Entre la generacion 1 a la 6 hay un total de {} Legentarios".format(cantidad))
```

```
Entre la generacion 1 a la 6 hay un total de 65 Legentarios
```

Punto #6

Devuelve el pokemon con más puntos de:

- Vida (hp)
- Ataque (attack)
- Defensa (defense)

> **En este caso usamos show para mostrar el que tiene mas puntos de una manera mas bonita, pero se puede usar .first() para obtener unicamente el primero**

In [47]:
```
from pyspark.sql.functions import desc

print("Pokemon con más puntos de vida")
all_pkmn.orderBy(desc("hp")).show(1)

print("Pokemon con más puntos de ataque")
all_pkmn.orderBy(desc("attack")).show(1)
```

```
print("Pokemon con más puntos de defensa")
all_pkmn.orderBy(desc("defense")).show(1)
```

```
Pokemon con más puntos de vida
+---+-------+------+------+-----+------+-------+------+------+-----+---------+
| id|   name|type_1|type_2|   hp|attack|defense|sp_atk|sp_def|speed|legendary|
+---+-------+------+------+-----+------+-------+------+------+-----+---------+
|242|Blissey|Normal|  null|255.0|  10.0|   10.0|  75.0| 135.0| 55.0|    false|
+---+-------+------+------+-----+------+-------+------+------+-----+---------+
only showing top 1 row

Pokemon con más puntos de ataque
+---+--------------------+-------+--------+-----+------+-------+------+------+-----+--
-------+
| id|                name| type_1|  type_2|   hp|attack|defense|sp_atk|sp_def|speed|le
gendary|
+---+--------------------+-------+--------+-----+------+-------+------+------+-----+--
-------+
|150|MewtwoMega Mewtwo X|Psychic|Fighting|106.0| 190.0|  100.0| 154.0| 100.0|130.0|
true|
+---+--------------------+-------+--------+-----+------+-------+------+------+-----+--
-------+
only showing top 1 row

Pokemon con más puntos de defensa
+---+--------------------+------+------+----+------+-------+------+------+-----+------
---+
| id|                name|type_1|type_2|  hp|attack|defense|sp_atk|sp_def|speed|legend
ary|
+---+--------------------+------+------+----+------+-------+------+------+-----+------
---+
|208|SteelixMega Steelix| Steel|Ground|75.0| 125.0|  230.0|  55.0|  95.0| 30.0|    fa
lse|
+---+--------------------+------+------+----+------+-------+------+------+-----+------
---+
only showing top 1 row
```

Punto #7

Devuelve el pokemon con menos puntos de:

- Ataque especial (sp_attack)
- Defensa especial (sp_defense)
- Velocidad (speed)

> **En este caso usamos show para mostrar el que tiene mas puntos de una manera mas bonita, pero se puede usar .first() para obtener unicamente el primero**

In [50]:
```
print("Pokemon con menos puntos de ataque especial")
all_pkmn.orderBy("sp_atk").show(1)

print("Pokemon con menos puntos de defensa especial")
all_pkmn.orderBy("sp_def").show(1)
```

```
print("Pokemon con menos puntos de valocidad")
all_pkmn.orderBy("speed").show(1)
```

```
Pokemon con menos puntos de ataque especial
+---+------+------+------+----+------+-------+------+------+-----+---------+
| id|  name|type_1|type_2|  hp|attack|defense|sp_atk|sp_def|speed|legendary|
+---+------+------+------+----+------+-------+------+------+-----+---------+
|438|Bonsly|  Rock|  null|50.0|  80.0|   95.0|  10.0|  45.0| 10.0|    false|
+---+------+------+------+----+------+-------+------+------+-----+---------+
only showing top 1 row

Pokemon con menos puntos de defensa especial
+---+--------+------+------+----+------+-------+------+------+-----+---------+
| id|    name|type_1|type_2|  hp|attack|defense|sp_atk|sp_def|speed|legendary|
+---+--------+------+------+----+------+-------+------+------+-----+---------+
|318|Carvanha| Water|  Dark|45.0|  90.0|   20.0|  65.0|  20.0| 65.0|    false|
+---+--------+------+------+----+------+-------+------+------+-----+---------+
only showing top 1 row

Pokemon con menos puntos de valocidad
+---+-------+------+------+----+------+-------+------+------+-----+---------+
| id|   name|type_1|type_2|  hp|attack|defense|sp_atk|sp_def|speed|legendary|
+---+-------+------+------+----+------+-------+------+------+-----+---------+
|213|Shuckle|   Bug|  Rock|20.0|  10.0|  230.0|  10.0| 230.0|  5.0|    false|
+---+-------+------+------+----+------+-------+------+------+-----+---------+
only showing top 1 row
```

Punto #8

Realiza un Join de la información de los pokemon y sus ventajas

In [74]:
```
from pyspark.sql.types import StructType, StructField, StringType

schema = StructType([
    StructField("tipo", StringType(), True),
    StructField("ventaja_1", StringType(), True),
    StructField("ventaja_2", StringType(), True),
    StructField("ventaja_3", StringType(), True),
    StructField("ventaja_4", StringType(), True),
    StructField("ventaja_5", StringType(), True),
    StructField("desventaja_1", StringType(), True),
    StructField("desventaja_2", StringType(), True),
    StructField("desventaja_3", StringType(), True),
    StructField("desventaja_4", StringType(), True),
    StructField("desventaja_5", StringType(), True),
    StructField("sin_efecto", StringType(), True),
])

# Leer el archivo Excel con pandas
df_pandas = pd.read_excel("Data/ventajas.xlsx")

# Convertir el DataFrame de pandas a DataFrame de PySpark
df_spark = spark.createDataFrame(df_pandas, schema=schema)

# Realizar el Join

full_data = all_pkmn.join(df_spark, all_pkmn["type_1"] == df_spark["tipo"])
```

Punto #9

Del resultado final, devuelve toda la información del pokemon, pero solo la información de las primeras 2 ventajas, las primeras 2 desventajas y el campo “sin_efecto”

> **Por efectos practicos se mostrara una version resumida de la informacion del pokemon para que se pueda apreciar correctamente**

In [82]:
```
resultado = full_data.select("id", "name","type_1", "type_2", "hp","attack","defense",


full_data.select("id", "name","type_1" , "ventaja_1", "ventaja_2", "desventaja_1", "de
```

```
+---+--------------------+-------+---------+---------+------------+------------+-----
-----+
| id|                name| type_1|ventaja_1|ventaja_2|desventaja_1|desventaja_2|sin_e
fecto|
+---+--------------------+-------+---------+---------+------------+------------+-----
-----+
| 63|                Abra|Psychic| Luchador|   Veneno|    Psíquico|       Acero| Sini
estro|
|617|            Accelgor|    Bug|   Planta| Psíquico|       Fuego|       Lucha|   Ni
nguno|
|681|AegislashBlade Forme|  Steel|    Hielo|     Roca|       Fuego|        Agua|   Ni
nguno|
|681|AegislashShield F...|  Steel|    Hielo|     Roca|       Fuego|        Agua|   Ni
nguno|
|142|          Aerodactyl|   Rock|    Fuego|    Hielo|       Lucha|      Tierra|   Ni
nguno|
|142|AerodactylMega Ae...|   Rock|    Fuego|    Hielo|       Lucha|      Tierra|   Ni
nguno|
|306|              Aggron|  Steel|    Hielo|     Roca|       Fuego|        Agua|   Ni
nguno|
|306|   AggronMega Aggron|  Steel|    Hielo|     Roca|       Fuego|        Agua|   Ni
nguno|
|190|               Aipom| Normal|  Ninguna|  Ninguna|        Roca|       Acero|  Fan
tasma|
| 65|            Alakazam|Psychic| Luchador|   Veneno|    Psíquico|       Acero| Sini
estro|
+---+--------------------+-------+---------+---------+------------+------------+-----
-----+
only showing top 10 rows
```

Punto #10

Escribe el resultado en un archivo parquet pkmn_result__info.parquet

In [88]:
```
# Guardar como CSV
resultado.write.csv("pkmn_result__info.csv", sep=",", header=True, mode="overwrite")

# Guardar como parquet
resultado.write.format("parquet").save("pkmn_result__info.parquet")
```