Importing Datasets

In [43]:
```
import pandas as pd                                 #1.import pandas library
df = pd.read_csv('pokemon_data.csv')                #2.reading the file from the directory
df
```

Out[43]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...
795	719	Diancie	Rock	Fairy	50	100	150	100	150	50	6	True
796	719	DiancieMega Diancie	Rock	Fairy	50	160	110	160	110	110	6	True
797	720	HoopaHoopa Confined	Psychic	Ghost	80	110	60	150	130	70	6	True
798	720	HoopaHoopa Unbound	Psychic	Dark	80	160	60	170	130	80	6	True
799	721	Volcanion	Fire	Water	80	110	120	130	90	70	6	True

800 rows × 12 columns

In [40]:
```
pwd    #shows where the directory is working and must save the file there
```

Out[40]:
```
'C:\\Users\\shiridi'
```

Reading the Data

In [44]:
```
df.columns
#Read Headers
```

Out[44]:
```
Index(['#', 'Name', 'Type 1', 'Type 2', 'HP', 'Attack', 'Defense', 'Sp. Atk',
       'Sp. Def', 'Speed', 'Generation', 'Legendary'],
      dtype='object')
```

In [45]:
```
df['Name']
#Reading each Column
```

Out[45]:
```
0                  Bulbasaur
1                    Ivysaur
2                   Venusaur
3      VenusaurMega Venusaur
4                 Charmander
               ...
795                  Diancie
796      DiancieMega Diancie
797      HoopaHoopa Confined
798       HoopaHoopa Unbound
799                Volcanion
Name: Name, Length: 800, dtype: object
```

In [2]:
```
df[['Name','HP']]
#Reading multiple columns
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-30c931c10aa3> in <module>
----> 1 dw = df[['Name','HP']]
      2 dw           #Reading multiple columns

NameError: name 'df' is not defined
```

In [54]:
```
df.head(10)
#Reading no.of columns
```

Out[54]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	False
5	5	Charmeleon	Fire	NaN	58	64	58	80	65	80	1	False
6	6	Charizard	Fire	Flying	78	84	78	109	85	100	1	False
7	6	CharizardMega Charizard X	Fire	Dragon	78	130	111	130	85	100	1	False
8	6	CharizardMega Charizard Y	Fire	Flying	78	104	78	159	115	100	1	False
9	7	Squirtle	Water	NaN	44	48	65	50	64	43	1	False

Data Wrangling

The process of converting the raw data into another format in order to prepare the data for further analysis.

In [2]:
```
import pandas as pd
df = pd.read_csv('pokemon_data.csv')
dw = df.head(100)
dw
```

Out[2]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...
95	88	Grimer	Poison	NaN	80	80	50	40	50	25	1	False
96	89	Muk	Poison	NaN	105	105	75	65	100	50	1	False
97	90	Shellder	Water	NaN	30	65	100	45	25	40	1	False
98	91	Cloyster	Water	Ice	50	95	180	85	45	70	1	False
99	92	Gastly	Ghost	Poison	30	35	30	100	35	80	1	False

100 rows × 12 columns

In [13]:
```
dw['Attack'].mean()                     #to find a mean for a particular column
dw['HP'].mode()                         #to find a mode for a particular column
dw['Speed'].median()                    #to find a median for a particular column
dw['Defense'].std()                     #to find a standard deviation
dw['Type 2'].mode()
```

Out[13]:
```
0    Poison
dtype: object
```

In [6]:
```
import pandas as pd                                 #Handling with missing data
ff = pd.read_csv('weather_data.csv')
ff
```

Out[6]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [7]:
```
nff = ff.fillna(0)  #using fillna() and mentioning the value in the () the values can be cha
nged
nff
```

Out[7]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	0.0	9.0	Sunny
2	1/5/2017	28.0	0.0	Snow
3	1/6/2017	0.0	7.0	0
4	1/7/2017	32.0	0.0	Rain
5	1/8/2017	0.0	0.0	Sunny
6	1/9/2017	0.0	0.0	0
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [ ]:
```
#sometimes having 0 as nan is not a best guess so we need to come with someother values
#and i want to give a different values in each column
```

In [10]:
```
nff['windspeed'].median()
```

Out[10]:
```
6.0
```

In [11]:
```
nff['temperature'].median()
```

Out[11]:
```
28.0
```

In [12]:
```
nff['event'].mode()
```

Out[12]:
```
0    Sunny
dtype: object
```

In [20]:
```
ff
```

Out[20]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [21]:
```
new = ff.fillna({         #here in the dictionary we must give the values for each column
    'temperature':28.0,
    'windspeed':6.0,
    'event':'Sunny'
    })
new
```

Out[21]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	28.0	9.0	Sunny
2	1/5/2017	28.0	6.0	Snow
3	1/6/2017	28.0	7.0	Sunny
4	1/7/2017	32.0	6.0	Rain
5	1/8/2017	28.0	6.0	Sunny
6	1/9/2017	28.0	6.0	Sunny
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [ ]:
```
#and we have other two methods to replace the nan values   ffill&bfill
```

In [22]:
```
new2  = ff.fillna(method = 'ffill')       all the nan values are replaced with its forward v
alues
new2
```

Out[22]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	32.0	9.0	Sunny
2	1/5/2017	28.0	9.0	Snow
3	1/6/2017	28.0	7.0	Snow
4	1/7/2017	32.0	7.0	Rain
5	1/8/2017	32.0	7.0	Sunny
6	1/9/2017	32.0	7.0	Sunny
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [23]:
```
new3 = ff.fillna(method = 'bfill')   #bfill is used to fill nan values with its bacward valu
es
new3
```

Out[23]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	28.0	9.0	Sunny
2	1/5/2017	28.0	7.0	Snow
3	1/6/2017	32.0	7.0	Rain
4	1/7/2017	32.0	8.0	Rain
5	1/8/2017	34.0	8.0	Sunny
6	1/9/2017	34.0	8.0	Cloudy
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [25]:
```
ff
```

Out[25]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [26]:
```
dff = ff.dropna()              #drops all the rows which have nan values
dff
```

Out[26]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [27]:
```
dff1 = ff.dropna(how='all')    #dropping only the column which have all nan values
dff1
```

Out[27]:

	day	temperature	windspeed	event
0	1/1/2017	32.0	6.0	Rain
1	1/4/2017	NaN	9.0	Sunny
2	1/5/2017	28.0	NaN	Snow
3	1/6/2017	NaN	7.0	NaN
4	1/7/2017	32.0	NaN	Rain
5	1/8/2017	NaN	NaN	Sunny
6	1/9/2017	NaN	NaN	NaN
7	1/10/2017	34.0	8.0	Cloudy
8	1/11/2017	40.0	12.0	Sunny

In [4]:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-dcbf514dfbb5> in <module>
----> 1 ff

NameError: name 'ff' is not defined
```

In [ ]: