Day02 - Session 2 - Data Manupulation using Pandas

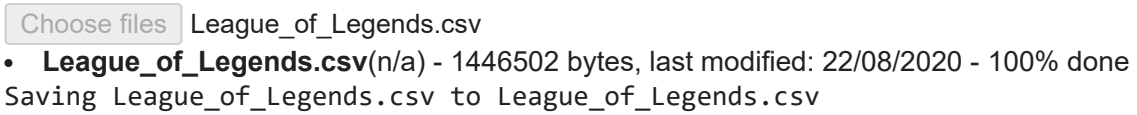
Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures.

Note:

1. First Clean the Evironment (Go to "Kernel" Menu --> "Restart & Clean Output"
2. To execute the code --> Click on a cell and press cntrl + enter key

```
from google.colab import files
```

```
uploaded = files.upload()
```



- **League_of_Legends.csv**(n/a) - 1446502 bytes, last modified: 22/08/2020 - 100% done

```
Saving League_of_Legends.csv to League_of_Legends.csv
```

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Working with Pandas

1. Import pandas library

```
#This command imports all the methods related to pandas.

import pandas as pd
```

2 Let's start with Series

Series is a one-dimensional labeled array

2.1 A Series is created with data from 1 to 9

```
import pandas as pd

a = [1, 3, 5, 7, 9, 2, 4, 6, 8]
a1 = pd.Series(a)

print(a1)
```

2.2 A Series has been created with Data along with it's Index

```
import pandas as pd

a1 = [1,3,5,7,9,2,4,6,8]
a2 = ['a','b','c','d','e','f','g','h','i']
a3 = pd.Series(a1,a2)

print(a3)
a3['b']
```

2.3 Creating a series with the help of a dictionary

```
import pandas as pd

dict1 = {'Oranges':3, 'Apples':4, 'Mangoes':2, 'Banana':12}
dict2 = pd.Series(dict1)

print (dict2)
print (type(dict2))
```

2.4 Creating a series with the help of Nested List

```
import pandas as pd

Array1 = [[1,3,5],[2,4,6]]
Array2 = pd.Series(Array1)

print (Array2)
```

2 DataFrames

DataFrames are 2 dimensional data structure which are defined in PANDAS which has rows and columns.

2.1 Creating a data frame with dictionary

```
import pandas as pd

Data = {'Age':[23,33,12,45],'Name':['Rahul','John','Robert','Sneha']}
Data1 = pd.DataFrame(Data)

print(Data1)
```

2.2 Creating a data frame with lists

```
import pandas as pd

Data2 = [[4,1900],[3,1600],[2,1100],[1,850]]
Data3 = pd.DataFrame(Data2, columns = ['No_of_Bedrooms','Square_Feet'])

print (Data3)
```

2.3 Assigning indexes within a data frame

```
import pandas as pd

Data4 = {'Name':['Ankit','Rishitha','Karthik','Vishnu'],'Marks':[78,67,98,56]}
Data5 = pd.DataFrame(Data4,index = ['Rank 2','Rank 3','Rank 1','Rank 4'])

print (Data5)
```

2.4 Creating dataframes from list of dictionaries

```
import pandas as pd

Data6 = [{'A':65,'B':66,'C':67},{'A':97,'B':98,'C':99}]
Data7 = pd.DataFrame(Data6)

print (Data7)
```

2.5 Creating a dataframe with the help of timestamp and categorical.

```
import numpy as np
import pandas as pd

Data8 = pd.DataFrame({'A':1, 'B':pd.Timestamp('20190305'),'C':np.array([3]*4)
                     , 'D' : pd.Categorical(["Test","Train","Car","Bike"])
                     , 'E':'Hello, Welcome!'})
print(Data8)
```

3 Working with data file (csv)

3.1 Read csv file

```
import pandas as pd

LOL = pd.read_csv('League_of_Legends.csv')
LOL
```

	gameId	blueWins	blueWardsPlaced	blueWardsDestroyed	blueFirstBlood	blu
0	4519157822	0	28	2	1	
1	4523371949	0	12	1	0	
2	4521474530	0	15	0	0	
3	4524384067	0	43	1	0	
4	4436033771	0	75	4	0	
...	...	...	...	...	...	...
9874	4527873286	1	17	2	1	
9875	4527797466	1	54	0	0	
9876	4527713716	0	23	1	0	
9877	4527628313	0	14	4	1	
9878	4523772935	1	18	0	1	

9879 rows × 40 columns

3.2 Get the dimention of the dataset

```
LOL.shape
```

```
(9879, 40)
```

3.3 Top 5 rows of the Data Set

```
LOL.head()
```

	gameId	blueWins	blueWardsPlaced	blueWardsDestroyed	blueFirstBlood	blueKi
0	4519157822	0	28	2	1	
1	4523371949	0	12	1	0	
2	4521474530	0	15	0	0	
3	4524384067	0	43	1	0	
4	4436033771	0	75	4	0	

3.4 Bottom 5 rows of the Data Set

```
LOL.tail()
```

	gameId	blueWins	blueWardsPlaced	blueWardsDestroyed	blueFirstBlood	blu
9874	4527873286	1	17	2	1	
9875	4527797466	1	54	0	0	
9876	4527713716	0	23	1	0	
9877	4527628313	0	14	4	1	
9878	4523772935	1	18	0	1	

3.5 Get all column names of the Data Set

```
LOL.columns
```

```
Index(['gameId', 'blueWins', 'blueWardsPlaced', 'blueWardsDestroyed',
       'blueFirstBlood', 'blueKills', 'blueDeaths', 'blueAssists',
       'blueEliteMonsters', 'blueDragons', 'blueHeralds',
       'blueTowersDestroyed', 'blueTotalGold', 'blueAvgLevel',
       'blueTotalExperience', 'blueTotalMinionsKilled',
       'blueTotalJungleMinionsKilled', 'blueGoldDiff', 'blueExperienceDiff',
       'blueCSPerMin', 'blueGoldPerMin', 'redWardsPlaced', 'redWardsDestroyed',
       'redFirstBlood', 'redKills', 'redDeaths', 'redAssists',
       'redEliteMonsters', 'redDragons', 'redHeralds', 'redTowersDestroyed',
       'redTotalGold', 'redAvgLevel', 'redTotalExperience',
       'redTotalMinionsKilled', 'redTotalJungleMinionsKilled', 'redGoldDiff',
       'redExperienceDiff', 'redCSPerMin', 'redGoldPerMin'],
      dtype='object')
```

3.6 Get the statistical summary of the data

```
LOL.describe()
```

	gameId	blueWins	blueWardsPlaced	blueWardsDestroyed	blueFirstBlood
count	9.879000e+03	9879.000000	9879.000000	9879.000000	9879.000000
mean	4.500084e+09	0.499038	22.288288	2.824881	0.504808
std	2.757328e+07	0.500024	18.019177	2.174998	0.500002
min	4.295358e+09	0.000000	5.000000	0.000000	0.000000
25%	4.483301e+09	0.000000	14.000000	1.000000	0.000000
50%	4.510920e+09	0.000000	16.000000	3.000000	1.000000
75%	4.521733e+09	1.000000	20.000000	4.000000	1.000000
max	4.527991e+09	1.000000	250.000000	27.000000	1.000000

3.7 Get the information related to the Data Frame

```
LOL.info(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9879 entries, 0 to 9878
Data columns (total 40 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   gameId                        9879 non-null   int64
 1   blueWins                      9879 non-null   int64
 2   blueWardsPlaced               9879 non-null   int64
 3   blueWardsDestroyed            9879 non-null   int64
```

```
 4   blueFirstBlood                9879 non-null   int64
 5   blueKills                     9879 non-null   int64
 6   blueDeaths                    9879 non-null   int64
 7   blueAssists                   9879 non-null   int64
 8   blueEliteMonsters             9879 non-null   int64
 9   blueDragons                   9879 non-null   int64
 10  blueHeralds                   9879 non-null   int64
 11  blueTowersDestroyed           9879 non-null   int64
 12  blueTotalGold                 9879 non-null   int64
 13  blueAvgLevel                  9879 non-null   float64
 14  blueTotalExperience           9879 non-null   int64
 15  blueTotalMinionsKilled        9879 non-null   int64
 16  blueTotalJungleMinionsKilled  9879 non-null   int64
 17  blueGoldDiff                  9879 non-null   int64
 18  blueExperienceDiff            9879 non-null   int64
 19  blueCSPerMin                  9879 non-null   float64
 20  blueGoldPerMin                9879 non-null   float64
 21  redWardsPlaced                9879 non-null   int64
 22  redWardsDestroyed             9879 non-null   int64
 23  redFirstBlood                 9879 non-null   int64
 24  redKills                      9879 non-null   int64
 25  redDeaths                     9879 non-null   int64
 26  redAssists                    9879 non-null   int64
 27  redEliteMonsters              9879 non-null   int64
 28  redDragons                    9879 non-null   int64
 29  redHeralds                    9879 non-null   int64
 30  redTowersDestroyed            9879 non-null   int64
 31  redTotalGold                  9879 non-null   int64
 32  redAvgLevel                   9879 non-null   float64
 33  redTotalExperience            9879 non-null   int64
 34  redTotalMinionsKilled         9879 non-null   int64
 35  redTotalJungleMinionsKilled   9879 non-null   int64
 36  redGoldDiff                   9879 non-null   int64
 37  redExperienceDiff             9879 non-null   int64
 38  redCSPerMin                   9879 non-null   float64
 39  redGoldPerMin                 9879 non-null   float64
dtypes: float64(6), int64(34)
memory usage: 3.0 MB
```

3.8 Transposing the Dataframe

```
LOL.T.tail(15)
```

	0	1	2	3	4	5	6
redDeaths	9.0	5.0	7.0	4.0	6.0	5.0	7.0
redAssists	8.0	2.0	14.0	10.0	7.0	2.0	9.0
redEliteMonsters	0.0	2.0	0.0	0.0	1.0	0.0	0.0
redDragons	0.0	1.0	0.0	0.0	1.0	0.0	0.0
redHeralds	0.0	1.0	0.0	0.0	0.0	0.0	0.0
redTowersDestroyed	0.0	1.0	0.0	0.0	0.0	0.0	0.0
redTotalGold	16567.0	17620.0	17285.0	16478.0	17404.0	15201.0	14463.0
redAvgLevel	6.8	6.8	6.8	7.0	7.0	7.0	6.4
redTotalExperience	17047.0	17438.0	17254.0	17961.0	18313.0	18060.0	15404.0
redTotalMinionsKilled	197.0	240.0	203.0	235.0	225.0	221.0	164.0

3.9 Get columns using column names

redGoldDiff	-643.0	2968.0	1172.0	1321.0	1004.0	-698.0	-2411.0

```
LOL.loc[:,['gameId','redKills','blueKills']]
```

	gameId	redKills	blueKills
0	4519157822	6	9
1	4523371949	5	5
2	4521474530	11	7
3	4524384067	5	4
4	4436033771	6	6
...	...	...	...
9874	4527873286	4	7
9875	4527797466	4	6
9876	4527713716	7	6
9877	4527628313	3	2
9878	4523772935	6	6

9879 rows × 3 columns

3.10 Get columns using position

```
LOL.iloc[0:2,2:5]
```

	blueWardsPlaced	blueWardsDestroyed	blueFirstBlood
0	28	2	1
1	12	1	0

3.11 Get the mean of the all the columns present in the dataset

```
LOL.mean().tail(10)
```

```
redTowersDestroyed               0.043021
redTotalGold                 16489.041401
redAvgLevel                      6.925316
redTotalExperience           17961.730438
redTotalMinionsKilled          217.349226
redTotalJungleMinionsKilled     51.313088
redGoldDiff                    -14.414111
redExperienceDiff               33.620306
redCSPerMin                     21.734923
redGoldPerMin                 1648.904140
dtype: float64
```

3.12 Get the correlation of the all the columns present in the dataset

```
LOL.corr().head(10)
```

	gameId	blueWins	blueWardsPlaced	blueWardsDestroyed	blueFi
gameId	1.000000	0.000985	0.005361	-0.012057	-
blueWins	0.000985	1.000000	0.000087	0.044247	
blueWardsPlaced	0.005361	0.000087	1.000000	0.034447	
blueWardsDestroyed	-0.012057	0.044247	0.034447	1.000000	
blueFirstBlood	-0.011577	0.201769	0.003228	0.017717	
blueKills	-0.038993	0.337358	0.018138	0.033748	
blueDeaths	-0.013160	-0.339297	-0.002612	-0.073182	-
blueAssists	-0.023329	0.276685	0.033217	0.067793	
blueEliteMonsters	0.016599	0.221944	0.019892	0.041700	
blueDragons	0.008962	0.213768	0.017676	0.040504	

3.13 Get the maximum value of the data set present in each column

```
LOL.max().head(10)
```

```
gameId                4.527991e+09
blueWins              1.000000e+00
blueWardsPlaced       2.500000e+02
blueWardsDestroyed    2.700000e+01
blueFirstBlood        1.000000e+00
blueKills             2.200000e+01
blueDeaths            2.200000e+01
blueAssists           2.900000e+01
blueEliteMonsters     2.000000e+00
blueDragons           1.000000e+00
dtype: float64
```

3.14 Get the minimum value of the dataset of each column

```
LOL.min().tail(12)
```

```
redDragons                       0.0
redHeralds                       0.0
redTowersDestroyed               0.0
redTotalGold                 11212.0
redAvgLevel                      4.8
redTotalExperience           10465.0
redTotalMinionsKilled          107.0
redTotalJungleMinionsKilled      4.0
redGoldDiff                -11467.0
redExperienceDiff           -8348.0
redCSPerMin                    10.7
redGoldPerMin                1121.2
dtype: float64
```

3.15 Get the median of the Dataset

```
LOL.median().head(13)
```

```
gameId                4.510920e+09
blueWins              0.000000e+00
blueWardsPlaced       1.600000e+01
blueWardsDestroyed    3.000000e+00
blueFirstBlood        1.000000e+00
blueKills             6.000000e+00
blueDeaths            6.000000e+00
```

```
blueAssists            6.000000e+00
blueEliteMonsters      0.000000e+00
blueDragons            0.000000e+00
blueHeralds            0.000000e+00
blueTowersDestroyed    0.000000e+00
blueTotalGold          1.639800e+04
dtype: float64
```

3.16 Get the standard deviation of the dataset

```
LOL.std().head(10)
```

```
gameId                2.757328e+07
blueWins              5.000244e-01
blueWardsPlaced       1.801918e+01
blueWardsDestroyed    2.174998e+00
blueFirstBlood        5.000022e-01
blueKills             3.011028e+00
blueDeaths            2.933818e+00
blueAssists           4.064520e+00
blueEliteMonsters     6.255265e-01
blueDragons           4.805974e-01
dtype: float64
```

3.17 Append the dataset with the same dataset

```
print(LOL.shape)

LOL_temp = LOL.append(LOL)

print(LOL_temp.shape)
```

```
(9879, 40)
(19758, 40)
```

3.18 Drop the duplicates present in the dataset.

```
print(LOL.shape)

LOL_temp = LOL_temp.drop_duplicates()

print(LOL_temp.shape)
```

```
(9879, 40)
```

```
(9879, 40)
```

3.19 IsNull: This returns true or false depending on the status of the cell

```
import pandas as pd

LOL = pd.read_csv('League_of_Legends.csv')
LOL.isnull()
```

	gameId	blueWins	blueWardsPlaced	blueWardsDestroyed	blueFirstBlood	blueKil
0	False	False	False	False	False	Fal
1	False	False	False	False	False	Fal
2	False	False	False	False	False	Fal
3	False	False	False	False	False	Fal
4	False	False	False	False	False	Fal
...	...	...	...	...	...	
9874	False	False	False	False	False	Fal
9875	False	False	False	False	False	Fal
9876	False	False	False	False	False	Fal
9877	False	False	False	False	False	Fal
9878	False	False	False	False	False	Fal

9879 rows × 40 columns

3.20 Aggregate of all the values which are null

```
LOL.isnull().sum().head(5)
```

```
gameId               0
blueWins             0
blueWardsPlaced      0
blueWardsDestroyed   0
blueFirstBlood       0
dtype: int64
```

3.21 Drop NA values (delete rows)

```
import pandas as pd
import numpy as np
```

```
import numpy as np

Data9 = pd.DataFrame({"Name":["Iron-Man","Wonder-Woman","Avengers"],
                      "House":["Marvel","DC Comics","Marvel"],
                      "Start":[pd.NaT,pd.Timestamp("2017-05-15"),pd.NaT]})

Data9
```

	Name	House	Start
0	Iron-Man	Marvel	NaT
1	Wonder-Woman	DC Comics	2017-05-15
2	Avengers	Marvel	NaT

```
Data9.dropna()
```

	Name	House	Start
1	Wonder-Woman	DC Comics	2017-05-15

3.22 Drop the columns where there are null values

```
Data9.dropna(axis = 'columns')
```

	Name	House
0	Iron-Man	Marvel
1	Wonder-Woman	DC Comics
2	Avengers	Marvel

3.23 Drop the entire row and column if ALL THE VALUES are null

```
Data9.dropna(how = 'all')
```

	Name	House	Start

3.24 Drop the null values where they are present

1	Wonder-Woman	DC Comics	2017-05-15

```
Data9.dropna(how = 'any')
```

	Name	House	Start
1	Wonder-Woman	DC Comics	2017-05-15

3.25 fill the null values with '0'

```
import pandas as pd
import numpy as np

Data10 = pd.DataFrame([[3,np.nan,4,2],[5,2,np.nan,9],
                       [np.nan,np.nan,7,np.nan],[4,np.nan,5,np.nan]]
                      ,columns=list('PQRS'))
Data10
```

	P	Q	R	S
0	3.0	NaN	4.0	2.0
1	5.0	2.0	NaN	9.0
2	NaN	NaN	7.0	NaN
3	4.0	NaN	5.0	NaN

```
Data10.fillna(0)
```

	P	Q	R	S
0	3.0	0.0	4.0	2.0
1	5.0	2.0	0.0	9.0
2	0.0	0.0	7.0	0.0
3	4.0	0.0	5.0	0.0

3.26 Replace Values

```
Replace Values = {'P':10,'O':11,'R':12,'S':13}
```

```
Data10.fillna(Replace_Values)
```

	P	Q	R	S
0	3.0	11.0	4.0	2.0
1	5.0	2.0	12.0	9.0
2	10.0	11.0	7.0	13.0
3	4.0	11.0	5.0	13.0

3.27 Fill null values only once which are specified by the user

```
Data10.fillna(Replace_Values, limit = 1)
```

	P	Q	R	S
0	3.0	11.0	4.0	2.0
1	5.0	2.0	12.0	9.0
2	10.0	NaN	7.0	13.0
3	4.0	NaN	5.0	NaN

```
Data10
```

	P	Q	R	S
0	3.0	NaN	4.0	2.0
1	5.0	2.0	NaN	9.0
2	NaN	NaN	7.0	NaN
3	4.0	NaN	5.0	NaN

3.28 Calculated the mean of column (ignore NA)

```
Mean1 = Data10['P'].mean()
Mean1
```

```
4.0
```

3.29 Filled the missing values with the calculated mean

```
Data10['P'].fillna(Mean1,inplace= True)
```

```
Data10
```

	P	Q	R	S
0	3.0	NaN	4.0	2.0
1	5.0	2.0	NaN	9.0
2	4.0	NaN	7.0	NaN
3	4.0	NaN	5.0	NaN

3.30 Describe the dataset

```
Data10.describe()
```

	P	Q	R	S
count	4.000000	1.0	3.000000	2.000000
mean	4.000000	2.0	5.333333	5.500000
std	0.816497	NaN	1.527525	4.949747
min	3.000000	2.0	4.000000	2.000000
25%	3.750000	2.0	4.500000	3.750000
50%	4.000000	2.0	5.000000	5.500000
75%	4.250000	2.0	6.000000	7.250000
max	5.000000	2.0	7.000000	9.000000

3.31 Describe the column of dataset

```
Data10['P'].describe()
```

```
count    4.000000
```

```
mean     4.000000
std      0.816497
min      3.000000
25%      3.750000
50%      4.000000
75%      4.250000
max      5.000000
Name: P, dtype: float64
```

3.32 Fill the missing valuse

```
Mean2 = Data10['Q'].mean()
Mean3 = Data10['R'].mean()
Mean4 = Data10['S'].mean()
```

```
Data10['Q'].fillna(Mean2,inplace= True)
Data10['R'].fillna(Mean3,inplace= True)
Data10['S'].fillna(Mean4,inplace= True)
```

```
Data10
```

	P	Q	R	S
0	3.0	2.0	4.000000	2.0
1	5.0	2.0	5.333333	9.0
2	4.0	2.0	7.000000	5.5
3	4.0	2.0	5.000000	5.5

3.33 Find the correlation between the columns

```
Data10.corr()
```

	P	Q	R	S
P	1.000000	NaN	0.436436	1.000000
Q	NaN	NaN	NaN	NaN
R	0.436436	NaN	1.000000	0.436436
S	1.000000	NaN	0.436436	1.000000

✓ 0s completed at 16:35