Task 2 - Data manipulation with Pandas

By Mallela Preethi

```python
import pandas as pd # Importing pandas library to do data manipulation
```

```python
from google.colab import files  # Uploading the data file
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving 01.Data Cleaning and Preprocessing.csv to 01.Data Cleaning and Preprocessing (1).csv
```

Reading the data

```
df = pd.read_csv('01.Data Cleaning and Preprocessing.csv') # reading the csv file in pandas
```

```
df.head() # explore the data
```

	Observation	Y-Kappa	ChipRate	BF-CMratio	BlowFlow	ChipLevel4	T-upperExt-2	T-lowerExt-2	UCZAA	WhiteFlow-4	...	SteamFlow-4
0	31-00:00	23.10	16.520	121.717	1177.607	169.805	358.282	329.545	1.443	599.253	...	67.122
1	31-01:00	27.60	16.810	79.022	1328.360	341.327	351.050	329.067	1.549	537.201	...	60.012
2	31-02:00	23.19	16.709	79.562	1329.407	239.161	350.022	329.260	1.600	549.611	...	61.304
3	31-03:00	23.60	16.478	81.011	1334.877	213.527	350.938	331.142	1.604	623.362	...	68.496
4	31-04:00	22.90	15.618	93.244	1334.168	243.131	351.640	332.709	NaN	638.672	...	70.022

5 rows × 23 columns

```
df.shape
```

```
(324, 23)
```

```python
df.info() # explore the data types
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 324 entries, 0 to 323
Data columns (total 23 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Observation     324 non-null    object
 1   Y-Kappa         324 non-null    float64
 2   ChipRate        319 non-null    float64
 3   BF-CMratio      307 non-null    float64
 4   BlowFlow        308 non-null    float64
 5   ChipLevel4      323 non-null    float64
 6   T-upperExt-2    322 non-null    float64
 7   T-lowerExt-2    322 non-null    float64
 8   UCZAA           299 non-null    float64
 9   WhiteFlow-4     323 non-null    float64
 10  AAWhiteSt-4     173 non-null    float64
 11  AA-Wood-4       323 non-null    float64
 12  ChipMoisture-4  323 non-null    float64
 13  SteamFlow-4     323 non-null    float64
 14  Lower-HeatT-3   322 non-null    float64
 15  Upper-HeatT-3   322 non-null    float64
 16  ChipMass-4      323 non-null    float64
 17  WeakLiquorF     323 non-null    float64
 18  BlackFlow-2     322 non-null    float64
 19  WeakWashF       323 non-null    float64
 20  SteamHeatF-3    322 non-null    float64
 21  T-Top-Chips-4   323 non-null    float64
 22  SulphidityL-4   173 non-null    float64
dtypes: float64(22), object(1)
memory usage: 58.3+ KB
```

Handling Missing Values

```python
df.isnull().sum()
```

```
Observation       0
Y-Kappa           0
ChipRate          5
BF-CMratio       17
BlowFlow         16
ChipLevel4        1
T-upperExt-2      2
T-lowerExt-2      2
UCZAA            25
WhiteFlow-4       1
AAWhiteSt-4     151
AA-Wood-4         1
ChipMoisture-4    1
SteamFlow-4       1
Lower-HeatT-3     2
Upper-HeatT-3     2
ChipMass-4        1
WeakLiquorF       1
BlackFlow-2       2
WeakWashF         1
SteamHeatF-3      2
T-Top-Chips-4     1
SulphidityL-4   151
dtype: int64
```

All are numerical valued columns and we can observe that there are some missing values in many of the columns let's explore the missing values and handle them.

Since there is less data which is of only 324 rows we can't afford eliminating the rows with missing values, So we fill the missing values with mean, ffill, bfill

```
[ ] # Fill with mean and ffill
    df['ChipRate'].fillna(df['ChipRate'].mean(), inplace=True)
    df['BF-CMratio'].fillna(df['BF-CMratio'].mean(), inplace=True)
    df['BlowFlow'].fillna(df['BlowFlow'].mean(), inplace=True)
    df['ChipLevel4 '].fillna(df['ChipLevel4 '].mean(), inplace=True)
    df['T-upperExt-2 '].fillna(df['T-upperExt-2 '].mean(), inplace=True)
    df['T-lowerExt-2  '].fillna(df['T-lowerExt-2  '].mean(), inplace=True)
    df['UCZAA'].fillna(df['UCZAA'].mean(), inplace=True)
    df['WhiteFlow-4 '].fillna(df['WhiteFlow-4 '].mean(), inplace=True)
    df['AA-Wood-4  '].fillna(df['AA-Wood-4  '].mean(), inplace=True)
    df['ChipMoisture-4 '].fillna(df['ChipMoisture-4 '].mean(), inplace=True)
    df['SteamFlow-4 '].fillna(df['SteamFlow-4 '].mean(), inplace=True)
    df['Lower-HeatT-3'].fillna(df['Lower-HeatT-3'].mean(), inplace=True)
    df['Upper-HeatT-3 '].fillna(df['Upper-HeatT-3 '].mean(), inplace=True)
    df['ChipMass-4 '].fillna(df['ChipMass-4 '].mean(), inplace=True)
    df['WeakLiquorF '].fillna(df['WeakLiquorF '].mean(), inplace=True)
    df['BlackFlow-2 '].fillna(df['BlackFlow-2 '].mean(), inplace=True)
    df['WeakWashF '].fillna(df['WeakWashF '].mean(), inplace=True)
    df['SteamHeatF-3 '].fillna(df['SteamHeatF-3 '].mean(), inplace=True)
    df['T-Top-Chips-4 '].fillna(df['T-Top-Chips-4 '].mean(), inplace=True)
```

```python
# Fill 'SulphidityL-4' and 'AAWhiteSt-4' with forward fill
df['SulphidityL-4 '].fillna(method='ffill', inplace=True)
df['SulphidityL-4 '].fillna(method='bfill', inplace=True) # to fill any null values after ffill

df['AAWhiteSt-4 '].fillna(method='ffill', inplace=True)
df['AAWhiteSt-4 '].fillna(method='bfill', inplace=True) # to fill any null values after ffill

# Verifying count of missing values
print(df.isnull().sum())
```

```
Observation        0
Y-Kappa            0
ChipRate           0
BF-CMratio         0
BlowFlow           0
ChipLevel4         0
T-upperExt-2       0
T-lowerExt-2       0
UCZAA              0
WhiteFlow-4        0
AAWhiteSt-4        0
AA-Wood-4          0
ChipMoisture-4     0
SteamFlow-4        0
Lower-HeatT-3      0
Upper-HeatT-3      0
ChipMass-4         0
WeakLiquorF        0
BlackFlow-2        0
WeakWashF          0
SteamHeatF-3       0
T-Top-Chips-4      0
SulphidityL-4      0
dtype: int64
```

Filtering data based on conditions

```
filtered_data = df[(df['ChipRate'] > 4) & (df['BlowFlow'] < 1000)]
filtered_data
```

	Observation	Y-Kappa	ChipRate	BF-CMratio	BlowFlow	ChipLevel4	T-upperExt-2	T-lowerExt-2	UCZAA	WhiteFlow-4	...
182	7-13:00	23.83	14.227	87.464456	0.000	220.074	352.981	323.718	1.416	594.970	
279	11-14:00	21.27	11.383	84.165000	981.920	342.858	352.315	322.292	1.553	592.539	
280	11-15:00	23.74	11.667	88.130000	990.724	349.088	349.697	311.997	1.555	579.875	
281	11-16:00	24.41	11.242	80.458000	954.092	365.583	342.403	302.669	1.556	546.509	
283	11-18:00	20.37	10.967	99.982000	998.153	302.251	344.295	305.080	1.604	471.537	

5 rows × 23 columns

```python
df[df['SulphidityL-4 '] > 31.7]
```

	Observation	Y-Kappa	ChipRate	BF-CMratio	BlowFlow	ChipLevel4	T-upperExt-2	T-lowerExt-2	UCZAA	WhiteFlow-4
80	3-07:00	26.50	16.300	75.411	1229.199	358.256	352.871	325.690	1.416	531.174
81	3-08:00	23.20	16.700	73.381	1225.454	288.327	353.400	325.761	1.532	546.814
82	3-09:00	24.20	16.458	75.625	1244.665	270.511	355.729	328.243	1.484	611.104
121	5-00:00	25.56	14.900	84.953	1289.167	373.726	358.028	326.809	1.265	568.074
122	5-01:00	24.29	15.175	85.006	1294.216	320.890	358.343	327.266	1.309	592.336
154	6-09:00	19.02	15.900	82.638	1292.604	218.707	362.036	330.491	1.546	695.383
155	6-10:00	16.12	15.642	81.822	1294.158	104.615	360.561	328.752	1.631	708.251

7 rows × 23 columns

Caluclating Summary Statistics

```
df.describe()
```

	Y-Kappa	ChipRate	BF-CMratio	BlowFlow	ChipLevel4	T-upperExt-2	T-lowerExt-2	UCZAA	WhiteFlow-4	AAWhiteSt-4	...
count	324.000000	324.000000	324.000000	324.000000	324.000000	324.000000	324.00000	324.000000	324.00000	324.000000	...
mean	20.635370	14.347937	87.464456	1237.837614	258.164483	356.904295	324.02018	1.492010	591.73226	6.142130	...
std	3.070036	1.487447	7.781774	98.070606	87.851143	9.180734	7.59777	0.101741	66.91253	0.080111	...
min	12.170000	9.983000	68.645000	0.000000	0.000000	339.168000	284.63300	1.182000	405.11100	5.890000	...
25%	18.382500	13.364750	82.156750	1194.525750	213.527000	350.291750	321.48600	1.436000	541.00225	6.092250	...
50%	20.845000	14.347937	87.253500	1254.658500	271.605500	356.901648	325.63850	1.492010	592.71700	6.137500	...
75%	23.032500	15.498250	92.123250	1288.628750	321.285000	362.104750	329.14700	1.555250	639.45775	6.200000	...
max	27.600000	16.958000	121.717000	1351.240000	419.014000	399.135000	337.01200	1.747000	731.39400	6.340000	...

8 rows × 22 columns

THANK YOU