PANDAS example 01

Import module

In [1]:

```
# for numpy and pandas
import numpy as np
import pandas as pd
```

Reading data

Import from Excel

supported data formats to import (Any one of these which selected by file extension)

file extension	data	etc
.xls	Office 97 ~ 2003 file format	data sheet
.xls	Office after 2003 file format	data sheet
.csv	Comma separated values	table
.tsv	Tab separated values	table with --sep value of '\t'
.json	JavaScript Object Notation	table of dict

Read excel(.xlsx) and save into variable `df` which is DataFrame (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html)

In [2]:

```
df = pd.read_excel('data/foo.xlsx')
df.tail(10)
```

Out[2]:

	Item ID	v	Item Desc 1	Item Desc 2	Wh	Qty	Price	MOQ	Lead Time	C.O.O
2551	HX18120500	2552	M12X1.25X50 961-8.8 ZN	NaN	0	100.0	NaN	NaN	NaN	NaN
2552	HX18120500	2553	M12X1.25X50 961-8.8 ZN	NaN	1	300.0	NaN	NaN	NaN	NaN
2553	HX18120500	2554	M12X1.25X50 961-8.8 ZN	NaN	2	60.0	NaN	NaN	NaN	NaN
2554	HX18120500	2555	M12X1.25X50 961-8.8 ZN	NaN	4	600.0	NaN	NaN	NaN	NaN
2555	HX18120500	2556	M12X1.25X50 961-8.8 ZN	NaN	6	330.0	NaN	NaN	NaN	NaN
2556	HX18120500	2557	M12X1.25X50 961-8.8 ZN	NaN	7	800.0	NaN	NaN	NaN	NaN
2557	HX18120500	2558	M12X1.25X50 961-8.8 ZN	NaN	MM	900.0	NaN	NaN	NaN	NaN
2558	NaN	2559	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2559	HX1820200	2560	M20X1.5X200 961-8.8	NaN	1	80.0	NaN	NaN	NaN	NaN
2560	HX1820200	2561	M20X1.5X200 961-8.8	NaN	MM	190.0	NaN	NaN	NaN	NaN

In [3]:

```
# Number of (rows,columns)
df.shape
```

Out[3]:

```
(2561, 10)
```

1. Preprocessing

1.1 Filtering

1.1.1 For column "Item Desc 1" filter out empty (NaN), refer Working with missing data (https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html)

df = df[df['Item Desc 1'].notnull()]

1.1.2 For Column "Qty" filter out less then 500

df = df[df['Qty'] >= 500.0]

1.1.3 For Column "Item ID" filter only starts with "HX", refer contains (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.str.contains.html?highlight=contains#pandas.Series.str.contains)

df = df[df['Item ID'].str.contains('^HX')]

1.1.4 For Column "Item ID" filter out starts with "HX"

df = df[~df['Item ID'].str.contains('^HX')]

In [4]:

```
df = df[df['Item Desc 1'].notnull()]
df.tail(10)
```

Out[4]:

	Item ID	v	Item Desc 1	Item Desc 2	Wh	Qty	Price	MOQ	Lead Time	C.O.O
2549	HX18120450	2550	M12X1.25X45 961-8.8 ZN	NaN	MM	140.0	NaN	NaN	NaN	NaN
2551	HX18120500	2552	M12X1.25X50 961-8.8 ZN	NaN	0	100.0	NaN	NaN	NaN	NaN
2552	HX18120500	2553	M12X1.25X50 961-8.8 ZN	NaN	1	300.0	NaN	NaN	NaN	NaN
2553	HX18120500	2554	M12X1.25X50 961-8.8 ZN	NaN	2	60.0	NaN	NaN	NaN	NaN
2554	HX18120500	2555	M12X1.25X50 961-8.8 ZN	NaN	4	600.0	NaN	NaN	NaN	NaN
2555	HX18120500	2556	M12X1.25X50 961-8.8 ZN	NaN	6	330.0	NaN	NaN	NaN	NaN
2556	HX18120500	2557	M12X1.25X50 961-8.8 ZN	NaN	7	800.0	NaN	NaN	NaN	NaN
2557	HX18120500	2558	M12X1.25X50 961-8.8 ZN	NaN	MM	900.0	NaN	NaN	NaN	NaN
2559	HX1820200	2560	M20X1.5X200 961-8.8	NaN	1	80.0	NaN	NaN	NaN	NaN
2560	HX1820200	2561	M20X1.5X200 961-8.8	NaN	MM	190.0	NaN	NaN	NaN	NaN

In [5]:

```
# After filtering Number of (rows,columns)
df.shape
```

Out[5]:

```
(1987, 10)
```

1.2 Replace Column

Replace "A " with "B" for column "Col"

`Col ::= 'A','B'`

means

df[`Col`] = df[`Col`].str.replace(`'A','B'`)

Refer [replace (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.replace.html?highlight=replace#pandas.Series.replace)](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.replace.html?highlight=replace#pandas.Series.replace)

Examples:

- df[`Col`] = df[`Col`].str.replace(`0, 5`)
- df[`Col`] = df[`Col`].str.replace(`[0, 1, 2, 3], 4`)
- df[`Col`] = df[`Col`].str.replace(`[0, 1, 2, 3], [4, 3, 2, 1]`)
- df[`Col`] = df[`Col`].str.replace(`'^foo|bar','foobar'`)
- df[`Col`] = df[`Col`].str.replace(`'^(\.\d+)',r'0\1'`) # .0002 => 0.0002

In [6]:

```
# get rid of "ISO "
df['Item Desc 1'] = df['Item Desc 1'].str.replace('ISO\s','')
df.tail(10)
```

Out[6]:

	Item ID	v	Item Desc 1	Item Desc 2	Wh	Qty	Price	MOQ	Lead Time	C.O.O
2549	HX18120450	2550	M12X1.25X45 961-8.8 ZN	NaN	MM	140.0	NaN	NaN	NaN	NaN
2551	HX18120500	2552	M12X1.25X50 961-8.8 ZN	NaN	0	100.0	NaN	NaN	NaN	NaN
2552	HX18120500	2553	M12X1.25X50 961-8.8 ZN	NaN	1	300.0	NaN	NaN	NaN	NaN
2553	HX18120500	2554	M12X1.25X50 961-8.8 ZN	NaN	2	60.0	NaN	NaN	NaN	NaN
2554	HX18120500	2555	M12X1.25X50 961-8.8 ZN	NaN	4	600.0	NaN	NaN	NaN	NaN
2555	HX18120500	2556	M12X1.25X50 961-8.8 ZN	NaN	6	330.0	NaN	NaN	NaN	NaN
2556	HX18120500	2557	M12X1.25X50 961-8.8 ZN	NaN	7	800.0	NaN	NaN	NaN	NaN
2557	HX18120500	2558	M12X1.25X50 961-8.8 ZN	NaN	MM	900.0	NaN	NaN	NaN	NaN
2559	HX1820200	2560	M20X1.5X200 961-8.8	NaN	1	80.0	NaN	NaN	NaN	NaN
2560	HX1820200	2561	M20X1.5X200 961-8.8	NaN	MM	190.0	NaN	NaN	NaN	NaN

In [7]:

```
# Number of (rows,columns)
df.shape
```

Out[7]:

```
(1987, 10)
```

1.3 Add columns from specific column extracing parts

- NewCol=lambda x: x['Col'].str.extract(r'^(\w+)')

Refer [str.extract (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.str.extract.html?highlight=extract#pandas.Series.str.extract)](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.str.extract.html?highlight=extract#pandas.Series.str.extract)

Add four columns from column "Item Desc 1"

Item Desc 1	IDS1	IDS2	IDS3	IDS4
M30X130 931-10.9 YZ	M30X130	931-10.9	YZ	
M12X60 ISO 4014-8.8 ZN				
M12X1.25X25 961-10.9 YZ .0002	M12X1.25X25	961-10.9	YZ	.0002

In [8]:

```
# extracting with regular expression in parentheses
df = df.assign(
    # Extracting starts with sequence of word, digits, '.' from 'Item Desc 1' an
d add IDS1
    IDS1=lambda x: x['Item Desc 1'].str.extract('^([\w\d\.]+)\s'),
    # Extracting sequence of digits, '-', '.' from 'Item Desc 1' and add IDS2
    IDS2=lambda x: x['Item Desc 1'].str.extract('\s([\d\-.]+)'),
    # Extracting sequence words after above matching from 'Item Desc 1' and add
 IDS3
    IDS3=lambda x: x['Item Desc 1'].str.extract('^[\w\d\.]+\s[\d\-.]+\s(\w+)'),
    # Extracting ends with sequence of digits, '.' from 'Item Desc 1' and add ID
S4
    IDS4=lambda x: x['Item Desc 1'].str.extract('\s([\d\.]+)$'),
)
df.tail(10)
```

Out[8]:

	Item ID	v	Item Desc 1	Item Desc 2	Wh	Qty	Price	MOQ	Lead Time	C.O.O	
2549	HX18120450	2550	M12X1.25X45 961-8.8 ZN	NaN	MM	140.0	NaN	NaN	NaN	NaN	M12X1.
2551	HX18120500	2552	M12X1.25X50 961-8.8 ZN	NaN	0	100.0	NaN	NaN	NaN	NaN	M12X1.
2552	HX18120500	2553	M12X1.25X50 961-8.8 ZN	NaN	1	300.0	NaN	NaN	NaN	NaN	M12X1.
2553	HX18120500	2554	M12X1.25X50 961-8.8 ZN	NaN	2	60.0	NaN	NaN	NaN	NaN	M12X1.
2554	HX18120500	2555	M12X1.25X50 961-8.8 ZN	NaN	4	600.0	NaN	NaN	NaN	NaN	M12X1.
2555	HX18120500	2556	M12X1.25X50 961-8.8 ZN	NaN	6	330.0	NaN	NaN	NaN	NaN	M12X1.
2556	HX18120500	2557	M12X1.25X50 961-8.8 ZN	NaN	7	800.0	NaN	NaN	NaN	NaN	M12X1.
2557	HX18120500	2558	M12X1.25X50 961-8.8 ZN	NaN	MM	900.0	NaN	NaN	NaN	NaN	M12X1.
2559	HX1820200	2560	M20X1.5X200 961-8.8	NaN	1	80.0	NaN	NaN	NaN	NaN	M20X1.
2560	HX1820200	2561	M20X1.5X200 961-8.8	NaN	MM	190.0	NaN	NaN	NaN	NaN	M20X1.

In [9]:

```
# after adding IDS1, IDS2, IDS3, IDS4 Number of (rows,columns)
df.shape
```

Out[9]:

```
(1987, 14)
```

Add columns

- Product Type : Sequence of digits before '-' in IDS2
- Grade : Sequence of digits after '-' in IDS2
- Diameter : Sequence of digites between 'M' and 'X' in IDS1
- Pitch : Sequence of digites between 'X' and 'X' in IDS1 (may be Null)
- Length : Sequence of digites after last 'X' in IDS1
- Etc : IDS4 and insert '0' in case of ".0002" as "0.0002"

In [10]:

```
df = df.assign(
    # Product Type : Sequence of digits before '-' in IDS2
    ProductType=lambda x: x['IDS2'].str.extract('^([\d]+)-'),
    # Grade : Sequence of digits after '-' in IDS2
    Grade=lambda x: x['IDS2'].str.extract('-([\d\.]+)$'),
    # Diameter : Sequence of digites between 'M' and 'X' in IDS1
    Diameter=lambda x: x['IDS1'].str.extract('^M(\d+)X'),
    # Pitch : Sequence of digites between 'X' and 'X' in IDS1 (may be Null)
    Pitch=lambda x: x['IDS1'].str.extract('X([\d\.]+)X'),
    # Length : Sequence of digites after last 'X' in IDS1
    Length=lambda x: x['IDS1'].str.extract('X([\d\.]+)$'),
    # Etc : IDS4 and insert '0' in case of ".0002" as "0.0002"
    Etc=lambda x: x['IDS4'].str.replace('^(\.\d+)',r'0\1'),
)
df.tail(10)
```

Out[10]:

	Item ID	v	Item Desc 1	Item Desc 2	Wh	Qty	Price	MOQ	Lead Time	C.O.O	
2549	HX18120450	2550	M12X1.25X45 961-8.8 ZN	NaN	MM	140.0	NaN	NaN	NaN	NaN	M12X1.
2551	HX18120500	2552	M12X1.25X50 961-8.8 ZN	NaN	0	100.0	NaN	NaN	NaN	NaN	M12X1.
2552	HX18120500	2553	M12X1.25X50 961-8.8 ZN	NaN	1	300.0	NaN	NaN	NaN	NaN	M12X1.
2553	HX18120500	2554	M12X1.25X50 961-8.8 ZN	NaN	2	60.0	NaN	NaN	NaN	NaN	M12X1.
2554	HX18120500	2555	M12X1.25X50 961-8.8 ZN	NaN	4	600.0	NaN	NaN	NaN	NaN	M12X1.
2555	HX18120500	2556	M12X1.25X50 961-8.8 ZN	NaN	6	330.0	NaN	NaN	NaN	NaN	M12X1.
2556	HX18120500	2557	M12X1.25X50 961-8.8 ZN	NaN	7	800.0	NaN	NaN	NaN	NaN	M12X1.
2557	HX18120500	2558	M12X1.25X50 961-8.8 ZN	NaN	MM	900.0	NaN	NaN	NaN	NaN	M12X1.
2559	HX1820200	2560	M20X1.5X200 961-8.8	NaN	1	80.0	NaN	NaN	NaN	NaN	M20X1.
2560	HX1820200	2561	M20X1.5X200 961-8.8	NaN	MM	190.0	NaN	NaN	NaN	NaN	M20X1.

In [11]:

```
# after adding 6 columns Number of (rows,columns)
df.shape
```

Out[11]:

```
(1987, 20)
```

Finish column

- Finish : IDS3 (may Null), replace according next table

Value					replaced
ZN	ZY	YZ	ZINC	=>	ZN
P&O	PHO	Phos		=>	PHO
PLAIN	Plain	PLN		=>	PLAIN

ZY, YZ, ZINC => ZN

P&O, Phos => PHO

Plain, PLN => PLAIN

In [12]:

```
# For each IDS3 column value do
# - if NaN then NaN, else
# - if 'ZY' then 'ZN', else
# - if 'YZ' then 'ZN', else
# - if 'ZINC' then 'ZN', else
# - if 'P&O' then 'PHO', else
# - if 'Phos' then 'PHO', else
# - if 'Plain' then 'PLAIN', else
# - if 'PLN' then 'PLAIN', else
# - set IDS3
df = df.assign(
    Finish=lambda x: np.where(x['IDS3'].isna(), np.nan,
                      np.where(x['IDS3'].str.contains('ZY'),'ZN',
                      np.where(x['IDS3'].str.contains('YZ'),'ZN',
                      np.where(x['IDS3'].str.contains('ZINC'),'ZN',
                      np.where(x['IDS3'].str.contains('P&O'),'PHO',
                      np.where(x['IDS3'].str.contains('Phos'),'PHO',
                      np.where(x['IDS3'].str.contains('Plain'),'PLAIN',
                      np.where(x['IDS3'].str.contains('PLN'),'PLAIN',x['IDS3'
]))))))))
)
df.tail(10)
```

Out[12]:

	Item ID	v	Item Desc 1	Item Desc 2	Wh	Qty	Price	MOQ	Lead Time	C.O.O	...	IDS
2549	HX18120450	2550	M12X1.25X45 961-8.8 ZN	NaN	MM	140.0	NaN	NaN	NaN	NaN	...	961 8.
2551	HX18120500	2552	M12X1.25X50 961-8.8 ZN	NaN	0	100.0	NaN	NaN	NaN	NaN	...	961 8.
2552	HX18120500	2553	M12X1.25X50 961-8.8 ZN	NaN	1	300.0	NaN	NaN	NaN	NaN	...	961 8.
2553	HX18120500	2554	M12X1.25X50 961-8.8 ZN	NaN	2	60.0	NaN	NaN	NaN	NaN	...	961 8.
2554	HX18120500	2555	M12X1.25X50 961-8.8 ZN	NaN	4	600.0	NaN	NaN	NaN	NaN	...	961 8.
2555	HX18120500	2556	M12X1.25X50 961-8.8 ZN	NaN	6	330.0	NaN	NaN	NaN	NaN	...	961 8.
2556	HX18120500	2557	M12X1.25X50 961-8.8 ZN	NaN	7	800.0	NaN	NaN	NaN	NaN	...	961 8.
2557	HX18120500	2558	M12X1.25X50 961-8.8 ZN	NaN	MM	900.0	NaN	NaN	NaN	NaN	...	961 8.
2559	HX1820200	2560	M20X1.5X200 961-8.8	NaN	1	80.0	NaN	NaN	NaN	NaN	...	961 8.
2560	HX1820200	2561	M20X1.5X200 961-8.8	NaN	MM	190.0	NaN	NaN	NaN	NaN	...	961 8.

10 rows × 21 columns

In [13]:

```
# after adding Finish, Number of (rows,columns)
df.shape
```

Out[13]:

```
(1987, 21)
```

1.4 drop columns

drop columns: IDS1, IDS2, IDS3, IDS4

In [14]:

```
df = df.drop(columns=['IDS1', 'IDS2', 'IDS3', 'IDS4'])
df.tail(10)
```

Out[14]:

	Item ID	v	Item Desc 1	Item Desc 2	Wh	Qty	Price	MOQ	Lead Time	C.O.O	Produc
2549	HX18120450	2550	M12X1.25X45 961-8.8 ZN	NaN	MM	140.0	NaN	NaN	NaN	NaN	
2551	HX18120500	2552	M12X1.25X50 961-8.8 ZN	NaN	0	100.0	NaN	NaN	NaN	NaN	
2552	HX18120500	2553	M12X1.25X50 961-8.8 ZN	NaN	1	300.0	NaN	NaN	NaN	NaN	
2553	HX18120500	2554	M12X1.25X50 961-8.8 ZN	NaN	2	60.0	NaN	NaN	NaN	NaN	
2554	HX18120500	2555	M12X1.25X50 961-8.8 ZN	NaN	4	600.0	NaN	NaN	NaN	NaN	
2555	HX18120500	2556	M12X1.25X50 961-8.8 ZN	NaN	6	330.0	NaN	NaN	NaN	NaN	
2556	HX18120500	2557	M12X1.25X50 961-8.8 ZN	NaN	7	800.0	NaN	NaN	NaN	NaN	
2557	HX18120500	2558	M12X1.25X50 961-8.8 ZN	NaN	MM	900.0	NaN	NaN	NaN	NaN	
2559	HX1820200	2560	M20X1.5X200 961-8.8	NaN	1	80.0	NaN	NaN	NaN	NaN	
2560	HX1820200	2561	M20X1.5X200 961-8.8	NaN	MM	190.0	NaN	NaN	NaN	NaN	

In [15]:

```
# after drop columns, Number of (rows,columns)
df.shape
```

Out[15]:

```
(1987, 17)
```

Export

supported data formats to export (Any one of these which selected by file extension)

file extension	data	etc
.xls	Office 97 ~ 2003 file format	data sheet
.xls	Office after 2003 file format	data sheet
.csv	Comma separated values	table
.tsv	Tab separated values	table with --sep value of '\t'
.json	JavaScript Object Notation	table of dict

In [16]:

```
# Export to Excel
df.to_excel('data/output.xlsx', index=False)
```

In [ ]:

```

```