```
## day objective
- pandas
### introduction o pandsas
- pandas is a phython library which palys a very imp role on data cleaning
```

In [1]:
```
##pip install pandas
##or
##conda install pandas
```

```
  File "<ipython-input-1-7bcaf9925e97>", line 1
    pip install pandas
              ^
SyntaxError: invalid syntax
```

In [2]:
```python
import pandas as pd
```

In [3]:
```python
pd.__version__
```

Out[3]:
```
'1.0.1'
```

```
In [4]: print(dir(pd))
```

```
['BooleanDtype', 'Categorical', 'CategoricalDtype', 'CategoricalIndex', 'DataFrame', 'DateOffset', 'DatetimeIndex', 'Da
tetimeTZDtype', 'ExcelFile', 'ExcelWriter', 'Float64Index', 'Grouper', 'HDFStore', 'Index', 'IndexSlice', 'Int16Dtype',
'Int32Dtype', 'Int64Dtype', 'Int64Index', 'Int8Dtype', 'Interval', 'IntervalDtype', 'IntervalIndex', 'MultiIndex', 'N
A', 'NaT', 'NamedAgg', 'Period', 'PeriodDtype', 'PeriodIndex', 'RangeIndex', 'Series', 'SparseDtype', 'StringDtype', 'T
imedelta', 'TimedeltaIndex', 'Timestamp', 'UInt16Dtype', 'UInt32Dtype', 'UInt64Dtype', 'UInt64Index', 'UInt8Dtype', '__
builtins__', '__cached__', '__doc__', '__docformat__', '__file__', '__getattr__', '__git_version__', '__loader__', '__n
ame__', '__package__', '__path__', '__spec__', '__version__', '_config', '_hashtable', '_lib', '_libs', '_np_version_un
der1p14', '_np_version_under1p15', '_np_version_under1p16', '_np_version_under1p17', '_np_version_under1p18', '_testin
g', '_tslib', '_typing', '_version', 'api', 'array', 'arrays', 'bdate_range', 'compat', 'concat', 'core', 'crosstab',
'cut', 'date_range', 'describe_option', 'errors', 'eval', 'factorize', 'get_dummies', 'get_option', 'infer_freq', 'inte
rval_range', 'io', 'isna', 'isnull', 'json_normalize', 'lreshape', 'melt', 'merge', 'merge_asof', 'merge_ordered', 'not
na', 'notnull', 'offsets', 'option_context', 'options', 'pandas', 'period_range', 'pivot', 'pivot_table', 'plotting',
'qcut', 'read_clipboard', 'read_csv', 'read_excel', 'read_feather', 'read_fwf', 'read_gbq', 'read_hdf', 'read_html', 'r
ead_json', 'read_orc', 'read_parquet', 'read_pickle', 'read_sas', 'read_spss', 'read_sql', 'read_sql_query', 'read_sql_
table', 'read_stata', 'read_table', 'reset_option', 'set_eng_float_format', 'set_option', 'show_versions', 'test', 'tes
ting', 'timedelta_range', 'to_datetime', 'to_numeric', 'to_pickle', 'to_timedelta', 'tseries', 'unique', 'util', 'value
_counts', 'wide_to_long']
```

1.series

- similar to Numpy 1-D array

In [5]:
```
help(pd.Series)
```

```
 |  __le__(self, other)
 |  
 |  __len__(self) -> int
 |      Return the length of the Series.
 |  
 |  __long__ = __int__(self)
 |  
 |  __lt__(self, other)
 |  
 |  __matmul__(self, other)
 |      Matrix multiplication using binary `@` operator in Python>=3.5.
 |  
 |  __mod__(left, right)
 |  
 |  __mul__(left, right)
 |  
 |  __ne__(self, other)
 |  
 |  __or__(self, other)
 |
```

- different ways to create a series
    - numpy
    - list
    - tuple
    - dictionary

In [6]:
```
pd.Series([12,13,55,97,8])
```

Out[6]:
```
0    12
1    13
2    55
3    97
4     8
dtype: int64
```

In [7]:
```
pd.Series([12,13.0,55,97,8])
# float is more complex then int
```

Out[7]:
```
0    12.0
1    13.0
2    55.0
3    97.0
4     8.0
dtype: float64
```

In [10]:
```
import numpy as np
pd.Series(np.array([12,56,9.99,100,87,5]),index=["a","b","c","d","e","f"])
```

Out[10]:
```
a     12.00
b     56.00
c      9.99
d    100.00
e     87.00
f      5.00
dtype: float64
```

In [11]:
```
dict1={"A":1000,"B":2000,"C":3000}
pd.Series(dict1)
```

Out[11]:
```
A    1000
B    2000
C    3000
dtype: int64
```

data frames

- data Frame is tabular format contains multiple no.of rows and columns
- pd.DataFrame()

In [13]:
```
d1=pd.DataFrame([100,200,39.0,77])
d1
```

Out[13]:

	0
0	100.0
1	200.0
2	39.0
3	77.0

In [17]:
```
d2=pd.DataFrame([["haritha",2020,10],["hemanjali",2021,11]],index=["A","B"],columns=["name","year","exp"])
d2
```

Out[17]:

	name	year	exp
A	haritha	2020	10
B	hemanjali	2021	11

In [20]:
```
d3=pd.DataFrame({"student":["haritha","hemanjali","harika"],
                 "trainers":["mounika","ruthu","lavanya"],
                 "Subjects":["DA","ML","AI"]},
                index=["I","II","III"])
d3
```

Out[20]:

	student	trainers	Subjects
I	haritha	mounika	DA
II	hemanjali	ruthu	ML
III	harika	lavanya	AI

Accesing Data

In [22]:
```
d3["student"]
```

Out[22]:
```
I         haritha
II      hemanjali
III        harika
Name: student, dtype: object
```

In [23]:
```
type(d3["student"])
```

Out[23]:
```
pandas.core.series.Series
```

In [24]:
```
type(d3)
```

Out[24]:
```
pandas.core.frame.DataFrame
```

In [27]:
```
d3["student"][1]
```

Out[27]:
```
'hemanjali'
```

In [28]:
```
d3[1]#error
#becaouse panda will get confuse whether '1' represents rows or columns?
```

...

indexing and Slicing

- selecting rows
  - dataframe[start:stop]
- selecting one column
  - dataframes["col_nmae"]
- selecting multiple columns
  - dataframe[["col_name"],["col_name2"]]

iloc and ioc

- iloc : acess default index values
    - dataframe.iloc[]
- ioc:user defined or index values
    - dataframeloc[]

In [29]:
```
d3.iloc[1]
```

Out[29]:
```
student     hemanjali
trainers        ruthu
Subjects           ML
Name: II, dtype: object
```

In [30]:
```
d3.loc["II"]
```

Out[30]:
```
student     hemanjali
trainers        ruthu
Subjects           ML
Name: II, dtype: object
```

In [33]:
```
d3.loc["II","Subjects"]
```

Out[33]:
```
'ML'
```

In [34]:
```
d3.iloc[1,2]
```

Out[34]:
```
'ML'
```

In [35]:
```
d3[["student","Subjects"]]
```

Out[35]:

	student	Subjects
I	haritha	DA
II	hemanjali	ML
III	harika	AI

In [36]:
```
df=pd.read_csv("https://raw.githubusercontent.com/nagamounika5/Datasets/master/Global%20Dataset/Market_Fact.csv")
```

In [37]:
```
df
```

Out[37]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.8100	0.01	23	-30.51	3.60	0.56
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.2700	0.01	13	4.56	0.93	0.54
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.6900	0.00	26	1148.90	2.50	0.59
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.34	14.30	0.37
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.87	26.30	0.38
...	...	...	...	...	...	...	...	...	...	...
8394	Ord_5353	Prod_4	SHP_7479	Cust_1798	2841.4395	0.08	28	374.63	7.69	0.59
8395	Ord_5411	Prod_6	SHP_7555	Cust_1798	127.1600	0.10	20	-74.03	6.92	0.37
8396	Ord_5388	Prod_6	SHP_7524	Cust_1798	243.0500	0.02	39	-70.85	5.35	0.40
8397	Ord_5348	Prod_15	SHP_7469	Cust_1798	3872.8700	0.03	23	565.34	30.00	0.62
8398	Ord_5459	Prod_6	SHP_7628	Cust_1798	603.6900	0.00	47	131.39	4.86	0.38

8399 rows × 10 columns

In [38]:
```
type(df)
```

Out[38]:
```
pandas.core.frame.DataFrame
```

filtering

In [42]:
```
df.head()#by default ,it displays first 5 rows
```

Out[42]:
```
    Ord_id   Prod_id   Ship_id    Cust_id    Sales  Discount  Order_Quantity   Profit  Shipping_Cost  Product_Base_Margin
0  Ord_5446  Prod_16  SHP_7609  Cust_1818   136.81      0.01              23   -30.51           3.60                 0.56
1  Ord_5406  Prod_13  SHP_7549  Cust_1818    42.27      0.01              13     4.56           0.93                 0.54
2  Ord_5446   Prod_4  SHP_7610  Cust_1818  4701.69      0.00              26  1148.90           2.50                 0.59
3  Ord_5456   Prod_6  SHP_7625  Cust_1818  2337.89      0.09              43   729.34          14.30                 0.37
4  Ord_5485  Prod_17  SHP_7664  Cust_1818  4233.15      0.08              35  1219.87          26.30                 0.38
```

In [40]:
```
df.head(10)
```

Out[40]:
```
    Ord_id   Prod_id   Ship_id    Cust_id      Sales  Discount  Order_Quantity   Profit  Shipping_Cost  Product_Base_Margin
0  Ord_5446  Prod_16  SHP_7609  Cust_1818   136.8100      0.01              23   -30.51           3.60                 0.56
1  Ord_5406  Prod_13  SHP_7549  Cust_1818    42.2700      0.01              13     4.56           0.93                 0.54
2  Ord_5446   Prod_4  SHP_7610  Cust_1818  4701.6900      0.00              26  1148.90           2.50                 0.59
3  Ord_5456   Prod_6  SHP_7625  Cust_1818  2337.8900      0.09              43   729.34          14.30                 0.37
4  Ord_5485  Prod_17  SHP_7664  Cust_1818  4233.1500      0.08              35  1219.87          26.30                 0.38
5  Ord_5446   Prod_6  SHP_7608  Cust_1818   164.0200      0.03              23   -47.64           6.15                 0.37
6    Ord_31  Prod_12    SHP_41    Cust_26    14.7600      0.01               5     1.32           0.50                 0.36
7  Ord_4725   Prod_4  SHP_6593  Cust_1641  3410.1575      0.10              48  1137.91           0.99                 0.55
8  Ord_4725  Prod_13  SHP_6593  Cust_1641   162.0000      0.01              33    45.84           0.71                 0.52
9  Ord_4725   Prod_6  SHP_6593  Cust_1641    57.2200      0.07               8   -27.72           6.60                 0.37
```

In [41]:
```
df.tail()
```

Out[41]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
8394	Ord_5353	Prod_4	SHP_7479	Cust_1798	2841.4395	0.08	28	374.63	7.69	0.59
8395	Ord_5411	Prod_6	SHP_7555	Cust_1798	127.1600	0.10	20	-74.03	6.92	0.37
8396	Ord_5388	Prod_6	SHP_7524	Cust_1798	243.0500	0.02	39	-70.85	5.35	0.40
8397	Ord_5348	Prod_15	SHP_7469	Cust_1798	3872.8700	0.03	23	565.34	30.00	0.62
8398	Ord_5459	Prod_6	SHP_7628	Cust_1798	603.6900	0.00	47	131.39	4.86	0.38

In [44]:
```
df.sample()#it displays only 1 random role
```

Out[44]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
3565	Ord_3053	Prod_2	SHP_4241	Cust_1137	1991.26	0.01	27	-528.09	60.0	0.41

In [46]:
```
df.sample(3)
```

Out[46]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
6435	Ord_3380	Prod_6	SHP_4686	Cust_1178	161.87	0.04	24	-31.27	5.11	0.37
7906	Ord_4000	Prod_13	SHP_5566	Cust_1374	17.06	0.03	5	-7.54	2.03	0.57
6108	Ord_4687	Prod_5	SHP_6540	Cust_1624	2447.65	0.00	50	1170.35	6.77	0.44

In [47]:
```
df.index
```

Out[47]:
```
RangeIndex(start=0, stop=8399, step=1)
```

In [48]:
```
df.columns
```

Out[48]:
```
Index(['Ord_id', 'Prod_id', 'Ship_id', 'Cust_id', 'Sales', 'Discount',
       'Order_Quantity', 'Profit', 'Shipping_Cost', 'Product_Base_Margin'],
      dtype='object')
```

In [50]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8399 entries, 0 to 8398
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Ord_id               8399 non-null   object 
 1   Prod_id              8399 non-null   object 
 2   Ship_id              8399 non-null   object 
 3   Cust_id              8399 non-null   object 
 4   Sales                8399 non-null   float64
 5   Discount             8399 non-null   float64
 6   Order_Quantity       8399 non-null   int64  
 7   Profit               8399 non-null   float64
 8   Shipping_Cost        8399 non-null   float64
 9   Product_Base_Margin  8336 non-null   float64
dtypes: float64(5), int64(1), object(4)
memory usage: 656.3+ KB
```

statistic

In [51]:
```
df.describe()#stastical details for every column
#we cant apply mean to string .....1st 3 col are obj type...so it is not applyied
```

Out[51]:

	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
count	8399.000000	8399.000000	8399.000000	8399.000000	8399.000000	8336.000000
mean	1775.878179	0.049671	25.571735	181.184424	12.838557	0.512513
std	3585.050525	0.031823	14.481071	1196.653371	17.264052	0.135589
min	2.240000	0.000000	1.000000	-14140.700000	0.490000	0.350000
25%	143.195000	0.020000	13.000000	-83.315000	3.300000	0.380000
50%	449.420000	0.050000	26.000000	-1.500000	6.070000	0.520000
75%	1709.320000	0.080000	38.000000	162.750000	13.990000	0.590000
max	89061.050000	0.250000	50.000000	27220.690000	164.730000	0.850000

In [56]:
```
df["Sales"]>30000
```

Out[56]:
```
0       False
1       False
2       False
3       False
4       False
        ...
8394    False
8395    False
8396    False
8397    False
8398    False
Name: Sales, Length: 8399, dtype: bool
```

In [55]:
```
df[ df["Sales"]>30000]
```

Out[55]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
1835	Ord_3875	Prod_17	SHP_5370	Cust_1351	41343.21	0.09	8	3852.19	24.49	0.39
2349	Ord_2373	Prod_14	SHP_3259	Cust_942	33367.85	0.01	9	3992.52	24.49	0.37
2738	Ord_3084	Prod_17	SHP_4279	Cust_1151	89061.05	0.00	13	27220.69	24.49	0.39
3784	Ord_2338	Prod_17	SHP_3207	Cust_932	45923.76	0.07	7	102.61	24.49	0.39

In [67]:

```python


df[(df["Sales"] > 25000) & (df["Sales"] < 30000)]
```

Out[67]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
385	Ord_3707	Prod_17	SHP_5136	Cust_1307	28359.40	0.05	49	14440.39	24.49	0.37
2216	Ord_4216	Prod_14	SHP_5881	Cust_1432	26126.92	0.04	42	9498.60	24.49	0.50
2253	Ord_3143	Prod_14	SHP_4362	Cust_1170	28664.52	0.09	50	13340.26	24.49	0.37
2259	Ord_1978	Prod_17	SHP_2703	Cust_725	25312.00	0.01	48	8788.81	16.63	0.59
2680	Ord_1963	Prod_11	SHP_2687	Cust_732	26622.55	0.08	49	3146.22	45.70	0.71
4124	Ord_2345	Prod_3	SHP_3218	Cust_937	25409.63	0.02	20	11535.28	19.99	0.35
4291	Ord_138	Prod_10	SHP_185	Cust_92	26133.39	0.04	30	-11053.60	44.55	0.62
4399	Ord_4614	Prod_14	SHP_6423	Cust_1571	29884.60	0.05	49	12748.86	24.49	0.44
4963	Ord_4161	Prod_17	SHP_5798	Cust_1421	25313.34	0.05	35	8612.11	16.06	0.56
5042	Ord_2425	Prod_14	SHP_3329	Cust_934	27820.34	0.08	48	11630.15	24.49	0.37
6037	Ord_3727	Prod_17	SHP_5171	Cust_1310	29186.49	0.05	38	11562.08	55.30	0.40
6245	Ord_997	Prod_14	SHP_1379	Cust_365	28761.52	0.04	8	285.11	24.49	0.37
6660	Ord_5425	Prod_14	SHP_7580	Cust_1799	27720.98	0.07	46	11984.40	24.49	0.37
6765	Ord_5186	Prod_17	SHP_7247	Cust_1763	26095.13	0.03	35	12606.81	55.30	0.40
7091	Ord_911	Prod_10	SHP_1255	Cust_302	28180.08	0.02	32	7513.88	44.55	0.62
7318	Ord_546	Prod_14	SHP_858	Cust_198	27875.54	0.00	46	-635.69	24.49	0.44
7547	Ord_3170	Prod_10	SHP_4400	Cust_1162	29345.27	0.03	34	7497.55	44.55	0.62
8046	Ord_825	Prod_14	SHP_1132	Cust_247	27663.92	0.05	8	-391.92	24.49	0.37
8217	Ord_3359	Prod_10	SHP_7245	Cust_1762	28389.14	0.07	33	7132.18	44.55	0.62

In [66]:
```
df[(df["Sales"] > 25000) & (df["Sales"] < 30000)]
```

Out[66]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
385	Ord_3707	Prod_17	SHP_5136	Cust_1307	28359.40	0.05	49	14440.39	24.49	0.37
2216	Ord_4216	Prod_14	SHP_5881	Cust_1432	26126.92	0.04	42	9498.60	24.49	0.50
2253	Ord_3143	Prod_14	SHP_4362	Cust_1170	28664.52	0.09	50	13340.26	24.49	0.37
2259	Ord_1978	Prod_17	SHP_2703	Cust_725	25312.00	0.01	48	8788.81	16.63	0.59
2680	Ord_1963	Prod_11	SHP_2687	Cust_732	26622.55	0.08	49	3146.22	45.70	0.71
4124	Ord_2345	Prod_3	SHP_3218	Cust_937	25409.63	0.02	20	11535.28	19.99	0.35
4291	Ord_138	Prod_10	SHP_185	Cust_92	26133.39	0.04	30	-11053.60	44.55	0.62
4399	Ord_4614	Prod_14	SHP_6423	Cust_1571	29884.60	0.05	49	12748.86	24.49	0.44
4963	Ord_4161	Prod_17	SHP_5798	Cust_1421	25313.34	0.05	35	8612.11	16.06	0.56
5042	Ord_2425	Prod_14	SHP_3329	Cust_934	27820.34	0.08	48	11630.15	24.49	0.37
6037	Ord_3727	Prod_17	SHP_5171	Cust_1310	29186.49	0.05	38	11562.08	55.30	0.40
6245	Ord_997	Prod_14	SHP_1379	Cust_365	28761.52	0.04	8	285.11	24.49	0.37
6660	Ord_5425	Prod_14	SHP_7580	Cust_1799	27720.98	0.07	46	11984.40	24.49	0.37
6765	Ord_5186	Prod_17	SHP_7247	Cust_1763	26095.13	0.03	35	12606.81	55.30	0.40
7091	Ord_911	Prod_10	SHP_1255	Cust_302	28180.08	0.02	32	7513.88	44.55	0.62
7318	Ord_546	Prod_14	SHP_858	Cust_198	27875.54	0.00	46	-635.69	24.49	0.44
7547	Ord_3170	Prod_10	SHP_4400	Cust_1162	29345.27	0.03	34	7497.55	44.55	0.62
8046	Ord_825	Prod_14	SHP_1132	Cust_247	27663.92	0.05	8	-391.92	24.49	0.37
8217	Ord_3359	Prod_10	SHP_7245	Cust_1762	28389.14	0.07	33	7132.18	44.55	0.62

In [68]:
```
df[(df["Sales"] > 25000) & (df["Sales"] < 30000)].shape
```

Out[68]:
```
(19, 10)
```

In [69]:
```
df[(df["Sales"] > 25000) & (df["Sales"] < 30000)].shape[0]
```

Out[69]:
```
19
```

In [70]:
```
df["Prod_id"].value_counts()
```

Out[70]:
```
Prod_6     1225
Prod_3      915
Prod_4      883
Prod_5      788
Prod_8      758
Prod_13     633
Prod_1      546
Prod_2      434
Prod_15     386
Prod_11     361
Prod_17     337
Prod_12     288
Prod_9      246
Prod_10     189
Prod_7      179
Prod_16     144
Prod_14      87
Name: Prod_id, dtype: int64
```

In [71]:
```
df["Prod_id"].value_counts().index
```

Out[71]:
```
Index(['Prod_6', 'Prod_3', 'Prod_4', 'Prod_5', 'Prod_8', 'Prod_13', 'Prod_1',
       'Prod_2', 'Prod_15', 'Prod_11', 'Prod_17', 'Prod_12', 'Prod_9',
       'Prod_10', 'Prod_7', 'Prod_16', 'Prod_14'],
      dtype='object')
```

In [72]:
```
df["Prod_id"]=="prod_14"
```

Out[72]:
```
0       False
1       False
2       False
3       False
4       False
        ...
8394    False
8395    False
8396    False
8397    False
8398    False
Name: Prod_id, Length: 8399, dtype: bool
```

In [75]:
```
df[df["Prod_id"]=="prod_14"]
```

Out[75]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin

In [76]:
```
df[df["Prod_id"]=="prod_14"].shape
```

Out[76]:
```
(0, 10)
```

data cleaning using pandas

working onduplicates

heandling missing values

In [81]:
```
d4=pd.DataFrame({"student":["haritha","hemanjali","harika","haritha"],
                 "trainers":["mounika","ruthu","lavanya","mounika"],
                 "Subjects":["DA","ML","AI","DA"]})
d4
```

Out[81]:

	student	trainers	Subjects
0	haritha	mounika	DA
1	hemanjali	ruthu	ML
2	harika	lavanya	AI
3	haritha	mounika	DA

In [82]:
```
d4.duplicated()
```

Out[82]:
```
0    False
1    False
2    False
3     True
dtype: bool
```

In [84]:
```
d4[d4.duplicated()]
```

Out[84]:

	student	trainers	Subjects
3	haritha	mounika	DA

In [85]:
```
#remove duplicate values
d4.drop_duplicates()
```

Out[85]:

	student	trainers	Subjects
0	haritha	mounika	DA
1	hemanjali	ruthu	ML
2	harika	lavanya	AI

In [86]:
```
d4
```

Out[86]:

	student	trainers	Subjects
0	haritha	mounika	DA
1	hemanjali	ruthu	ML
2	harika	lavanya	AI
3	haritha	mounika	DA

In [87]:
```
d4.drop_duplicates(inplace=True)#true== original is changed
# by default it is false
```

In [88]:
```
d4
```

Out[88]:

	student	trainers	Subjects
0	haritha	mounika	DA
1	hemanjali	ruthu	ML
2	harika	lavanya	AI

In [ ]:
```

```