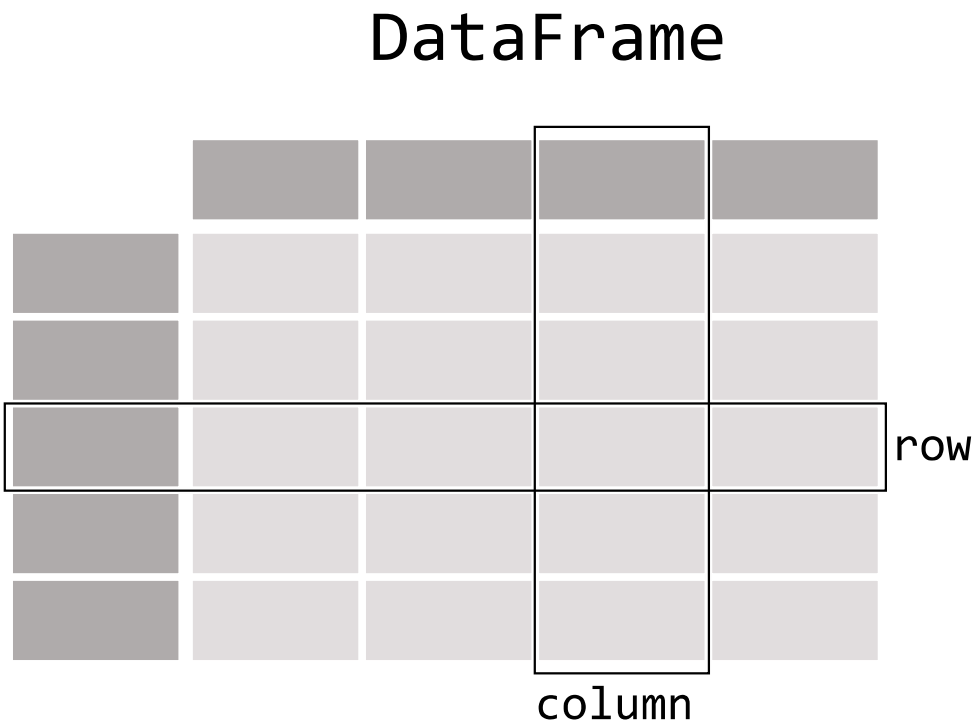
In [ ]:

```
import pandas as pd
```

Pandas

Pandas-stands for Panel data and is the core library for data manipulation and data analysis Pandas makes it simple to do many of the time consuming, repetitive tasks associated with working with data

pandas data table representation



Pandas Data Structures

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

1)Series: Pandas is a one-dimensional labeled array and capable of holding data of any type (integer, string, float, python objects, etc.)

Series is nothing but columns of an DataFrame

2)DataFrames:Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).

Basic example on creation of Series

In [57]:

```
import pandas as pd
s = pd.Series(['a','b','c'])
print(s)
```

```
0    a
1    b
2    c
dtype: object
```

In [58]:

```
import pandas as pd
s = pd.Series([1,2,3,4,5])
print(s)
```

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

Basic example on creation of DataFrame

In [93]:

```
import pandas as pd
d={
     'Name':['Tom', 'Joseph', 'Krish', 'John'],
     'Dep':['CSD','CSM','CSE','IT'],
     'Reg_No':['320','126','55','10']
    }
df=pd.DataFrame(d)
print(df)
```

```
     Name  Dep Reg_No
0     Tom  CSD    320
1  Joseph  CSM    126
2   Krish  CSE     55
3    John   IT     10
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

In [71]:

```
import pandas as pd
d1=[['Tom', 'CSD', 320],['Joseph','CSM',126],['krish','CSE',55],['Jhon','IT',10]]
df1=pd.DataFrame(d1,columns=['Name','Sec','Reg_No'],index=['A','B','C','D'])
print(df1)
```

```
     Name  Sec  Reg_No
A     Tom  CSD     320
B  Joseph  CSM     126
C   krish  CSE      55
D    Jhon   IT      10
```

General Methods

In below code NaN represents null value (or) None

head()---> to access 1st five rows of dataframe

tail()--->to access last five rows of dataframe

info()---->The info() method prints information about the DataFrame

In [64]:

```
df.head()
```

Out[64]:

	Name	Dep	Reg_No
0	Tom	CSD	320
1	Joseph	CSM	126
2	Krish	CSE	55
3	John	IT	10

In [68]:

```
#As it contains only 4 rows tail returns the last five rows which whole dataframe
df.tail()
```

Out[68]:

	Name	Dep	Reg_No
0	Tom	CSD	320
1	Joseph	CSM	126
2	Krish	CSE	55
3	John	IT	10

In [208]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Name    4 non-null      object
 1   Dep     4 non-null      object
 2   Reg_No  4 non-null      object
dtypes: object(3)
memory usage: 224.0+ bytes
```

Loading a CSV file

In [264]:

```
import pandas as pd
df=pd.read_csv("Downloads/stack-overflow-developer-survey-2019/survey_results_public.csv")
schema_df=pd.read_csv('Downloads/stack-overflow-developer-survey-2019/survey_results_schema
```

In [ ]:

```
pd.set_option('display.max_columns', 5)
pd.set_option('display.max_rows', 5)
```

In [265]:

```
df.head()
```

Out[265]:

	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenSource	Employment	Country	S
0	1	I am a student who is learning to code	Yes	Never	The quality of OSS and closed source software ...	Not employed, and not looking for work	United Kingdom	
1	2	I am a student who is learning to code	No	Less than once per year	The quality of OSS and closed source software ...	Not employed, but looking for work	Bosnia and Herzegovina	f
2	3	I am not primarily a developer, but I write co...	Yes	Never	The quality of OSS and closed source software ...	Employed full-time	Thailand	
3	4	I am a developer by profession	No	Never	The quality of OSS and closed source software ...	Employed full-time	United States	
4	5	I am a developer by profession	Yes	Once a month or more often	OSS is, on average, of HIGHER quality than pro...	Employed full-time	Ukraine	

5 rows × 85 columns

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

In [266]:

```
schema_df
```

Out[266]:

	Column	QuestionText
0	Respondent	Randomized respondent ID number (not in order ...
1	MainBranch	Which of the following options best describes ...
2	Hobbyist	Do you code as a hobby?
3	OpenSourcer	How often do you contribute to open source?
4	OpenSource	How do you feel about the quality of open sour...
...	...	...
80	Sexuality	Which of the following do you currently identi...
81	Ethnicity	Which of the following do you identify as? Ple...
82	Dependents	Do you have any dependents (e.g., children, el...
83	SurveyLength	How do you feel about the length of the survey...
84	SurveyEase	How easy or difficult was this survey to compl...

85 rows × 2 columns

Arithmetic and data alignment

In [251]:

```
import pandas as pd
df1 = pd.DataFrame(np.arange(9.).reshape((3,3)), columns=list('bcd'), index=['Ohio', "Texas
df2 = pd.DataFrame(np.arange(12.).reshape((4,3)), columns=list('bde'), index=["Uhah", 'Ohio
```

In [253]:

```
df1+df2
```

Out[253]:

	b	c	d	e
Colorado	NaN	NaN	NaN	NaN
Ohio	3.0	NaN	6.0	NaN
Oregon	NaN	NaN	NaN	NaN
Texas	9.0	NaN	12.0	NaN
Uhah	NaN	NaN	NaN	NaN

In [254]:

```
df1.add(df2, fill_value=0)
```

Out[254]:

	b	c	d	e
Colorado	6.0	7.0	8.0	NaN
Ohio	3.0	1.0	6.0	5.0
Oregon	9.0	NaN	10.0	11.0
Texas	9.0	4.0	12.0	8.0
Uhah	0.0	NaN	1.0	2.0

In [260]:

```
frame = pd.DataFrame(np.arange(12.).reshape((4,3)), columns=list('bde'), index=["Uhah", 'Oh

series2 = pd.Series(range(3), index=['b', 'e', 'f'])

frame + series2
```

Out[260]:

	b	d	e	f
Uhah	0.0	NaN	3.0	NaN
Ohio	3.0	NaN	6.0	NaN
Texas	6.0	NaN	9.0	NaN
Oregon	9.0	NaN	12.0	NaN

In [263]:

```
frame.sub(series2, axis=0)
```

Out[263]:

	b	d	e
Ohio	NaN	NaN	NaN
Oregon	NaN	NaN	NaN
Texas	NaN	NaN	NaN
Uhah	NaN	NaN	NaN
b	NaN	NaN	NaN
e	NaN	NaN	NaN
f	NaN	NaN	NaN

accessing columns

In [72]:

```
df['Name']
```

Out[72]:

```
0       Tom
1    Joseph
2     Krish
3      John
Name: Name, dtype: object
```

In [73]:

```
df[['Dep','Reg_No']]
```

Out[73]:

	Dep	Reg_No
0	CSD	320
1	CSM	126
2	CSE	55
3	IT	10

to assign particular value for a column

In [74]:

```
df['Name']="Krish"
print(df)
```

```
    Name  Dep Reg_No
0  Krish  CSD    320
1  Krish  CSM    126
2  Krish  CSE     55
3  Krish   IT     10
```

In [75]:

```
df.columns
```

Out[75]:

```
Index(['Name', 'Dep', 'Reg_No'], dtype='object')
```

Integer location (iloc) & Non Integer Location (loc)

loc syntax--> DataFrame.loc[ ]

Access a group of rows and columns by label(s) or a boolean array

iloc syntax--> DataFrame.iloc[ ]

Purely integer-location based indexing for selection by position

In [76]:

```
df.iloc[[1,2]]
```

Out[76]:

	Name	Dep	Reg_No
1	Krish	CSM	126
2	Krish	CSE	55

In [80]:

```
df.loc[[1,2]]
```

Out[80]:

	Name	Dep	Reg_No
1	Krish	CSM	126
2	Krish	CSE	55

In [77]:

```
df.iloc[[0, 1], 2]
```

Out[77]:

```
0    320
1    126
Name: Reg_No, dtype: object
```

In [78]:

```
df.loc[[0, 1], ['Name', 'Dep']]
```

Out[78]:

	Name	Dep
0	Krish	CSD
1	Krish	CSM

In [81]:

```
df.loc[[1,2],['Name','Dep','Reg_No']]
```

Out[81]:

	Name	Dep	Reg_No
1	Krish	CSM	126
2	Krish	CSE	55

In [ ]:

```
#index error
df.iloc[[1,2],['name','team','age']]
df.loc[[0,1],[1,2]]
```

set_index method

DataFrame.set_index(keys, drop=True, append=False, inplace=False,verify_integrity=False) Set the DataFrame index using existing columns.

Set the DataFrame index (row labels) using one or more existing columns or arrays (of the correct length). The index can replace the existing index or expand on it

In [82]:

```
df.set_index('Reg_No',inplace=True)
```

In [83]:

```
df
```

Out[83]:

	Name	Dep
Reg_No		
320	Krish	CSD
126	Krish	CSM
55	Krish	CSE
10	Krish	IT

In [84]:

```
df.index
```

Out[84]:

```
Index(['320', '126', '55', '10'], dtype='object', name='Reg_No')
```

In [85]:

```
df.iloc[0]
```

Out[85]:

```
Name    Krish
Dep       CSD
Name: 320, dtype: object
```

Reset_index

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

syntax=>DataFrame.reset_index(level=None,drop=False,inplace=False,col_level=0, col_fill='')

Reset the index of the DataFrame, and use the default one instead

In [86]:

```
df.reset_index(inplace=True)
df
```

Out[86]:

	Reg_No	Name	Dep
0	320	Krish	CSD
1	126	Krish	CSM
2	55	Krish	CSE
3	10	Krish	IT

Re-Indexing

Reindexing changes the row labels and column labels of a DataFrame. To reindex means to conform the data to match a given set of labels along a particular axis

In [87]:

```
new_ind=['A','B','C','D']
new_df=df.reindex(new_ind)
print(new_df)
```

```
   Reg_No Name  Dep
A     NaN  NaN  NaN
B     NaN  NaN  NaN
C     NaN  NaN  NaN
D     NaN  NaN  NaN
```

above *the values are set to NaN, which is the default behaviour when the new index is not the same as the old.*

Rename column by index

Pandas rename() method is used to rename any index, column or row.

Syntax-->*DataFrame.rename_axis(mapper=None, index=None, columns=None, axis=None, copy=True, inplace=False)*

In [90]:

```
import pandas as pd
df2 = pd.DataFrame({'a': [1, 2], 'b': [3, 4], 'c': [7, 8]})
mapping = {df.columns[0]: 'new0', df.columns[1]: 'new1'}
su = df2.rename(columns=mapping)
display(su)
```

	new0	new1	c
0	1	3	7
1	2	4	8

Dropping entries from axis

In [96]:

```
import pandas as pd
d={
     'Name':['Tom', 'Joseph', 'Krish', 'John'],
     'Dep':['CSD','CSM','CSE','IT'],
     'Reg_No':['320','126','55','10']
    }
df=pd.DataFrame(d)
print(df)
```

```
     Name  Dep Reg_No
0     Tom  CSD    320
1  Joseph  CSM    126
2   Krish  CSE     55
3    John   IT     10
```

In [97]:

```
#dropping row
ndf=df.drop([0,1],axis=0)
print(ndf)
```

```
    Name  Dep Reg_No
2  Krish  CSE     55
3   John   IT     10
```

In [98]:

```
#dropping column
n_df=df.drop(['Name','Dep'],axis=1)
print(n_df)
```

```
  Reg_No
0    320
1    126
2     55
3     10
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

In [99]:

```
#dropping both rows&columns
n_df=df.drop(index=2,columns='Reg_No')
print(n_df)
```

```
     Name  Dep
0     Tom  CSD
1  Joseph  CSM
3    John   IT
```

Filtering

filt is used to extract a particular rows & columns

Filtering: Using Conditionals to FilterRows and Columns

In [100]:

```
df['Name']=='Krish'
```

Out[100]:

```
0    False
1    False
2     True
3    False
Name: Name, dtype: bool
```

In [101]:

```
filt=(df['Dep']=='CSE')
```

In [102]:

```
df.loc[filt]
```

Out[102]:

	Name	Dep	Reg_No
2	Krish	CSE	55

In [103]:

```
df.loc[filt,'Dep']
```

Out[103]:

```
2    CSE
Name: Dep, dtype: object
```

In [104]:

```
df.loc[filt,['Dep','Reg_No']]
```

Out[104]:

	Dep	Reg_No
2	CSE	55

Can't use python built-in 'AND' and 'OR', so need to use symbols '&' and '|'

In [105]:

```
filt = (df['Name']=='Krish') & (df['Reg_No'] == '55')
df.loc[filt]
```

Out[105]:

	Name	Dep	Reg_No
2	Krish	CSE	55

In [106]:

```
filt=(df['Name']=='Joseph') | (df['Reg_No']=='55')
df.loc[filt]
```

Out[106]:

	Name	Dep	Reg_No
1	Joseph	CSM	126
2	Krish	CSE	55

In [107]:

```
df.loc[filt,['Name','Dep']]
```

Out[107]:

	Name	Dep
1	Joseph	CSM
2	Krish	CSE

In [108]:

```
#tilde simply it is the negation of the filter
df.loc[~filt]
```

Out[108]:

	Name	Dep	Reg_No
0	Tom	CSD	320
3	John	IT	10

DataFrame Alignments

syntax:DataFrame.align(other,join='outer',axis=None)

In [112]:

```
df = pd.DataFrame(
    [[1, 2, 3, 4], [6, 7, 8, 9]], columns=["D", "B", "E", "A"], index=[1, 2]
)
df2=pd.DataFrame([[10, 20, 30, 40], [60, 70, 80, 90], [600, 700, 800, 900]],
    columns=["A", "B", "C", "D"],
    index=[2, 3, 4],
)
df
```

Out[112]:

	D	B	E	A
1	1	2	3	4
2	6	7	8	9

In [113]:

```
df2
```

Out[113]:

	A	B	C	D
2	10	20	30	40
3	60	70	80	90
4	600	700	800	900

Align on columns:

In [114]:

```
x,y=df.align(df2,join='inner',axis=1)
x
```

Out[114]:

	D	B	A
1	1	2	4
2	6	7	9

In [115]:

```
y
```

Out[115]:

	D	B	A
2	40	20	10
3	90	70	60
4	900	700	600

align on index

In [116]:

```
x,y=df.align(df2,join='outer',axis=0)
x
```

Out[116]:

	D	B	E	A
1	1.0	2.0	3.0	4.0
2	6.0	7.0	8.0	9.0
3	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN

In [117]:

```
y
```

Out[117]:

	A	B	C	D
1	NaN	NaN	NaN	NaN
2	10.0	20.0	30.0	40.0
3	60.0	70.0	80.0	90.0
4	600.0	700.0	800.0	900.0

In [118]:

```
x,y=df.align(df2,join='outer',axis=None)
x
```

Out[118]:

	A	B	C	D	E
1	4.0	2.0	NaN	1.0	3.0
2	9.0	7.0	NaN	6.0	8.0
3	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN

In [119]:

```
y
```

Out[119]:

	A	B	C	D	E
1	NaN	NaN	NaN	NaN	NaN
2	10.0	20.0	30.0	40.0	NaN
3	60.0	70.0	80.0	90.0	NaN
4	600.0	700.0	800.0	900.0	NaN

Groupby

It creates a sub DataFrame from existing one

Due to this the descriptive statistical analysis will be easier on required Series

syntax=DataFrame.groupby(by=None,axis=0,level=None) by=mapping---> Used to determine the groups for the groupby

In [120]:

```
import pandas as pd

data = {
  'co2': [95, 90, 99, 104, 105, 94, 99, 104],
  'model': ['Citigo', 'Fabia', 'Fiesta', 'Rapid', 'Focus', 'Mondeo', 'Octavia', 'B-Max'],
  'car': ['Skoda', 'Skoda', 'Ford', 'Skoda', 'Ford', 'Ford', 'Skoda', 'Ford']
}

df5 = pd.DataFrame(data)

print(df5.groupby(["car"]).mean())
```

```
         co2
car
Ford   100.5
Skoda   97.0
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

In [121]:

```
l = [["a", 12, 12], [None, 12.3, 33.], ["b", 12.3, 123], ["a", 1, 1]]
df5 = pd.DataFrame(l, columns=["a", "b", "c"])
df5
```

Out[121]:

	a	b	c
0	a	12.0	12.0
1	None	12.3	33.0
2	b	12.3	123.0
3	a	1.0	1.0

In [122]:

```
df5.groupby(by="a").sum()
```

Out[122]:

	b	c
a		
a	13.0	13.0
b	12.3	123.0

In [123]:

```
df5.groupby(by="a", dropna=False).sum()
```

Out[123]:

	b	c
a		
a	13.0	13.0
b	12.3	123.0
NaN	12.3	33.0

Updating columns of DataFrame

In [157]:

```
import pandas as pd
d={
     'Name':['Tom', 'Joseph', 'Krish', 'John'],
     'Dep':['CSD','CSM','CSE','IT'],
     'Reg_No':['320','126','55','10']
    }
df=pd.DataFrame(d)
print(df)
```

```
     Name  Dep Reg_No
0     Tom  CSD    320
1  Joseph  CSM    126
2   Krish  CSE     55
3    John   IT     10
```

In [158]:

```
df2.columns=['names','sec','reg no']
df
```

Out[158]:

	Name	Dep	Reg_No
0	Tom	CSD	320
1	Joseph	CSM	126
2	Krish	CSE	55
3	John	IT	10

In [159]:

```
df.columns=[x.upper() for x in df.columns]
df
```

Out[159]:

	NAME	DEP	REG_NO
0	Tom	CSD	320
1	Joseph	CSM	126
2	Krish	CSE	55
3	John	IT	10

In [160]:

```
df.rename(columns={'NAMES':'S_NAMES','SEC':'D_NAME','REG NO':'ROLL_NO'},inplace=True)
df
```

Out[160]:

	NAME	DEP	REG_NO
0	Tom	CSD	320
1	Joseph	CSM	126
2	Krish	CSE	55
3	John	IT	10

In [161]:

```
df.loc[2, ['S_NAMES', 'ROLL_NO']] = ['Vijay', '046']
df
```

Out[161]:

	NAME	DEP	REG_NO	S_NAMES	ROLL_NO
0	Tom	CSD	320	NaN	NaN
1	Joseph	CSM	126	NaN	NaN
2	Krish	CSE	55	Vijay	046
3	John	IT	10	NaN	NaN

In [162]:

```
df.at[2,'D_NAME']='CSD'
df
```

Out[162]:

	NAME	DEP	REG_NO	S_NAMES	ROLL_NO	D_NAME
0	Tom	CSD	320	NaN	NaN	NaN
1	Joseph	CSM	126	NaN	NaN	NaN
2	Krish	CSE	55	Vijay	046	CSD
3	John	IT	10	NaN	NaN	NaN

Updating rows

In [191]:

```
import pandas as pd
d={
     'Name':['Tom', 'Joseph', 'Krish', 'John'],
     'Dep':['CSD','CSM','CSE','IT'],
     'Reg_No':['320','126','55','10']
    }
df=pd.DataFrame(d)
print(df)
```

```
     Name  Dep Reg_No
0     Tom  CSD    320
1  Joseph  CSM    126
2   Krish  CSE     55
3    John   IT     10
```

In [192]:

```
df.loc[2]
```

Out[192]:

```
Name      Krish
Dep         CSE
Reg_No       55
Name: 2, dtype: object
```

In [193]:

```
df.loc[0]=['Vijay','CSD','046']
df
```

Out[193]:

	Name	Dep	Reg_No
0	Vijay	CSD	046
1	Joseph	CSM	126
2	Krish	CSE	55
3	John	IT	10

In [194]:

```
filt = (df['Name'] == 'Vijay')
df[filt]['Reg_No']
```

Out[194]:

```
0    046
Name: Reg_No, dtype: object
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

In [195]:

```
filt = (df['Name'] == 'Vijay')
df.loc[filt,'Dep']="CSM"
df
```

Out[195]:

	Name	Dep	Reg_No
0	Vijay	CSM	046
1	Joseph	CSM	126
2	Krish	CSE	55
3	John	IT	10

Apply(): is used to apply a function along an axis of the DataFrame or on values of Series. map(): is used to substitute each value in a Series with another value

Applymap():is used to apply a function to a DataFrame elementwise

	DataFrame	Series
apply	✅	✅
map		✅
applymap	✅	

In [196]:

```
df['Name'].apply(len)
```

Out[196]:

```
0    5
1    6
2    5
3    4
Name: Name, dtype: int64
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

In [199]:

```
def update(Name):
    return Name.upper()
```

In [201]:

```
df['Name'].apply(update)
```

Out[201]:

```
0     VIJAY
1    JOSEPH
2     KRISH
3      JOHN
Name: Name, dtype: object
```

In [202]:

```
df
```

Out[202]:

	Name	Dep	Reg_No
0	VIJAY	CSM	046
1	JOSEPH	CSM	126
2	KRISH	CSE	55
3	JOHN	IT	10

In [170]:

```
df.apply(len, axis='columns')
```

Out[170]:

```
0    3
1    3
2    3
3    3
dtype: int64
```

In [171]:

```
len(df['Dep'])
```

Out[171]:

```
4
```

In [172]:

```
df.apply(lambda x: x.min())
```

Out[172]:

```
Name      John
Dep        CSE
Reg_No     046
dtype: object
```

In [173]:

```
df.applymap(len)
```

Out[173]:

	Name	Dep	Reg_No
0	5	3	3
1	6	3	3
2	5	3	2
3	4	2	2

In [174]:

```
df.applymap(str.lower)
```

Out[174]:

	Name	Dep	Reg_No
0	vijay	csm	046
1	joseph	csm	126
2	krish	cse	55
3	john	it	10

In [203]:

```
df['Dep'].map({'CSM': 'CSE', 'IT': 'ECE'})
```

Out[203]:

```
0    CSE
1    CSE
2    NaN
3    ECE
Name: Dep, dtype: object
```

In [204]:

```
df['Dep']=df['Dep'].replace({'CSM': 'CSE', 'IT': 'ECE'})
```

In [205]:

```
df
```

Out[205]:

	Name	Dep	Reg_No
0	VIJAY	CSE	046
1	JOSEPH	CSE	126
2	KRISH	CSE	55
3	JOHN	ECE	10

Add-Remove rows & Columns

In [214]:

```
import pandas as pd
```

In [217]:

```
data={
    'name_1':['vijay','Gopi','ram'],
    'name_2':['Ram','chand','kishn'],
     'email':['vijayram@gmail.com','gopichand@gmail.com','ramkrishn@gmail.com']
}
df=pd.DataFrame(data)
df
```

Out[217]:

	name_1	name_2	email
0	vijay	Ram	vijayram@gmail.com
1	Gopi	chand	gopichand@gmail.com
2	ram	kishn	ramkrishn@gmail.com

In [218]:

```
df['name_1']+ ' ' +df['name_2']
```

Out[218]:

```
0     vijay Ram
1    Gopi chand
2     ram kishn
dtype: object
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/STIX-Web/fontdata.js

In [220]:

```
df['full_name']=df['name_1']+ ' ' +df['name_2']
df
```

Out[220]:

	name_1	name_2	email	full_name
0	vijay	Ram	vijayram@gmail.com	vijay Ram
1	Gopi	chand	gopichand@gmail.com	Gopi chand
2	ram	kishn	ramkrishn@gmail.com	ram kishn

In [221]:

```
df.drop(columns=['name_1','name_2'],inplace=True)
df
```

Out[221]:

	email	full_name
0	vijayram@gmail.com	vijay Ram
1	gopichand@gmail.com	Gopi chand
2	ramkrishn@gmail.com	ram kishn

In [222]:

```
df['full_name'].str.split(" ",expand=True)
```

Out[222]:

	0	1
0	vijay	Ram
1	Gopi	chand
2	ram	kishn

In [223]:

```
df[['name_1', 'name_2']] = df['full_name'].str.split(' ', expand=True)
```

In [224]:

```
df
```

Out[224]:

	email	full_name	name_1	name_2
0	vijayram@gmail.com	vijay Ram	vijay	Ram
1	gopichand@gmail.com	Gopi chand	Gopi	chand
2	ramkrishn@gmail.com	ram kishn	ram	kishn

In [226]:

```
df.append({'name_1':'arjun'}, ignore_index=True)
```

```
C:\Users\rajap\AppData\Local\Temp\ipykernel_21844\2126306469.py:1: FutureWar
ning: The frame.append method is deprecated and will be removed from pandas
in a future version. Use pandas.concat instead.
  df.append({'name_1':'arjun'}, ignore_index=True)
```

Out[226]:

	email	full_name	name_1	name_2
0	vijayram@gmail.com	vijay Ram	vijay	Ram
1	gopichand@gmail.com	Gopi chand	Gopi	chand
2	ramkrishn@gmail.com	ram kishn	ram	kishn
3	NaN	NaN	arjun	NaN

In [229]:

```
data2 = {
    'first': ['Tony', 'Steve'],
    'last': ['Stark', 'Rogers'],
    'email': ['IronMan@avenge.com', 'Cap@avenge.com']
}
df2 = pd.DataFrame(data2)
df2
```

Out[229]:

	first	last	email
0	Tony	Stark	IronMan@avenge.com
1	Steve	Rogers	Cap@avenge.com

In [231]:

```
df=df.append(df2,ignore_index=True,sort=False)
df
```

```
C:\Users\rajap\AppData\Local\Temp\ipykernel_21844\1895738197.py:1: FutureWar
ning: The frame.append method is deprecated and will be removed from pandas
in a future version. Use pandas.concat instead.
  df=df.append(df2,ignore_index=True,sort=False)
```

Out[231]:

	email	full_name	name_1	name_2	first	last
0	vijayram@gmail.com	vijay Ram	vijay	Ram	NaN	NaN
1	gopichand@gmail.com	Gopi chand	Gopi	chand	NaN	NaN
2	ramkrishn@gmail.com	ram kishn	ram	kishn	NaN	NaN
3	IronMan@avenge.com	NaN	NaN	NaN	Tony	Stark
4	Cap@avenge.com	NaN	NaN	NaN	Steve	Rogers

In [232]:

```
df.drop(index=4)
```

Out[232]:

	email	full_name	name_1	name_2	first	last
0	vijayram@gmail.com	vijay Ram	vijay	Ram	NaN	NaN
1	gopichand@gmail.com	Gopi chand	Gopi	chand	NaN	NaN
2	ramkrishn@gmail.com	ram kishn	ram	kishn	NaN	NaN
3	IronMan@avenge.com	NaN	NaN	NaN	Tony	Stark

In [233]:

```
filt = df['last'] == 'Stark'
df.drop(index=df[filt].index)
```

Out[233]:

	email	full_name	name_1	name_2	first	last
0	vijayram@gmail.com	vijay Ram	vijay	Ram	NaN	NaN
1	gopichand@gmail.com	Gopi chand	Gopi	chand	NaN	NaN
2	ramkrishn@gmail.com	ram kishn	ram	kishn	NaN	NaN
4	Cap@avenge.com	NaN	NaN	NaN	Steve	Rogers

Sorting

In [238]:

```
import pandas as pd
data={
    'name_1':['vijay','Gopi','ram'],
    'name_2':['Ram','chand','kishn'],
     'email':['vijayram@gmail.com','gopichand@gmail.com','ramkrishn@gmail.com']
}
df=pd.DataFrame(data)
df
```

Out[238]:

	name_1	name_2	email
0	vijay	Ram	vijayram@gmail.com
1	Gopi	chand	gopichand@gmail.com
2	ram	kishn	ramkrishn@gmail.com

In [242]:

```
df.sort_values(by='name_2', ascending=False)
```

Out[242]:

	name_1	name_2	email
2	ram	kishn	ramkrishn@gmail.com
1	Gopi	chand	gopichand@gmail.com
0	vijay	Ram	vijayram@gmail.com

In [244]:

```
df.sort_values(by=['name_1', 'name_2'], ascending=[False,True],inplace=True)
df
```

Out[244]:

	name_1	name_2	email
0	vijay	Ram	vijayram@gmail.com
2	ram	kishn	ramkrishn@gmail.com
1	Gopi	chand	gopichand@gmail.com

In [245]:

```
df.sort_index()
```

Out[245]:

	name_1	name_2	email
0	vijay	Ram	vijayram@gmail.com
1	Gopi	chand	gopichand@gmail.com
2	ram	kishn	ramkrishn@gmail.com

In [246]:

```
df['name_1'].sort_values()
```

Out[246]:

```
1     Gopi
2      ram
0    vijay
Name: name_1, dtype: object
```