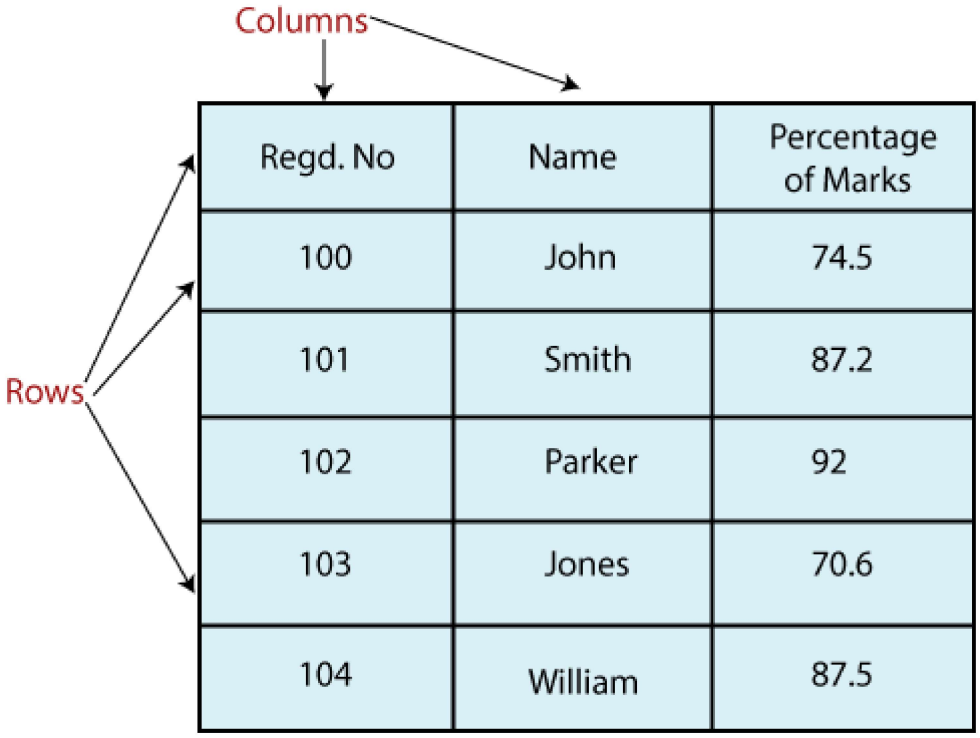
In [ ]:
```
'''
What is Pandas?
Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

Why Use Pandas?
Pandas allows us to analyze big data and make conclusions based on statistical theorie

Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science.


Pandas : Powerful for data analysis

Pandas stands for “Python Data Analysis Library”
Data wrangling –also called data cleaning, data remediation,
or data munging

'''
```



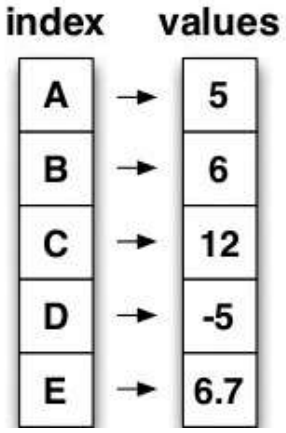
In [1]:
```
#!pip install pandas
import pandas as pd
import numpy as np
```

```
Requirement already satisfied: pandas in c:\users\kiran\appdata\local\programs\python
\python311\lib\site-packages (1.5.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\kiran\appdata\local
\programs\python\python311\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\kiran\appdata\local\programs
\python\python311\lib\site-packages (from pandas) (2022.6)
Requirement already satisfied: numpy>=1.21.0 in c:\users\kiran\appdata\local\programs
\python\python311\lib\site-packages (from pandas) (1.23.4)
Requirement already satisfied: six>=1.5 in c:\users\kiran\appdata\local\programs\pyth
on\python311\lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
[notice] A new release of pip is available: 23.0 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [2]:
```
'''
What is a Series?
A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.
'''
s1 = pd.Series([23,24,25,12,'32'])
s1
```

Out[2]:
```
0    23
1    24
2    25
3    12
4    32
dtype: object
```

Series



In [3]:
```
s2 = pd.Series([23,45,67,12,34],index = ['a','b','c','d','e'])
s2
```

Out[3]:
```
a    23
b    45
c    67
d    12
e    34
dtype: int64
```

In [4]:
```
s3 = pd.Series([23,45,67,12,34],index = ['a','b','c','d','e'],dtype = 'float')
s3
```

Out[4]:
```
a    23.0
b    45.0
c    67.0
d    12.0
e    34.0
dtype: float64
```

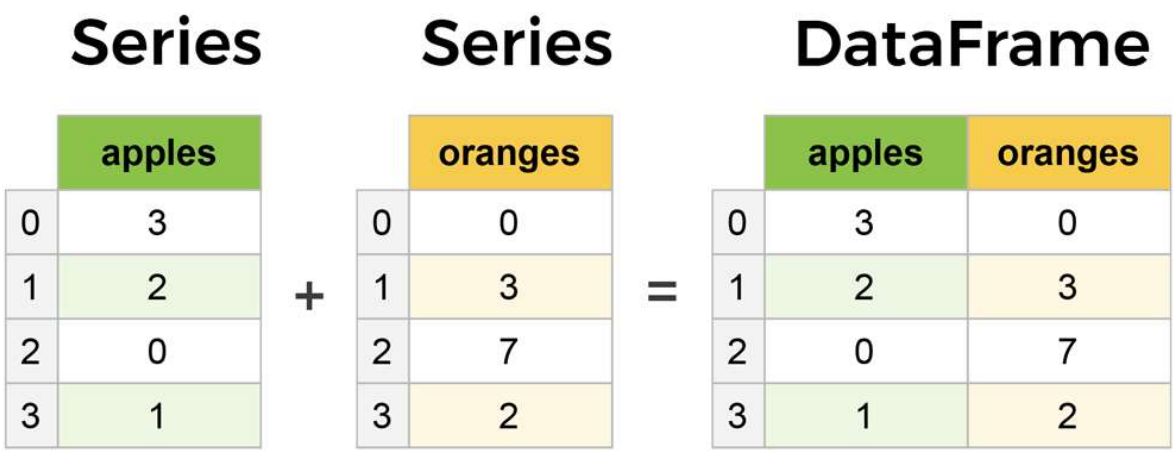
creation of Series using dictionary

In [5]:
```
s4 = pd.Series({'e':65,'f':43,'c':45})
s4
```

Out[5]:
```
e    65
f    43
c    45
dtype: int64
```

DataFrame

What is a DataFrame? A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.



In [6]:
```
d1 = pd.DataFrame([43,54,65,76])
d1
```

Out[6]:

	0
0	43
1	54
2	65
3	76

In [7]:
```
d2 = pd.DataFrame([[2,3,4],[4,5,6],[1,2,3]])
d2
```

Out[7]:

	0	1	2
0	2	3	4
1	4	5	6
2	1	2	3

In [8]:
```
d2 = pd.DataFrame(s2)  #converting series into dataframe
d2
```

Out[8]:

	0
a	23
b	45
c	67
d	12
e	34

In [9]:
```
d3  = pd.DataFrame([[2,3,4],[4,5,6],[1,2,3]],columns =['a','b','c'])
d3
```

Out[9]:

	a	b	c
0	2	3	4
1	4	5	6
2	1	2	3

In [10]:
```
d3  = pd.DataFrame([[2,3,4],[4,5,6],[1,2,3]],columns =['a','b','c'],index = ['x','y',
d3
```

Out[10]:

	a	b	c
x	2	3	4
y	4	5	6
z	1	2	3

creating DataFrame from list of dictionaries

In [11]:
```
dic = [{'alex':1,'joe':2},{'ema':5,'dora':10,'alice':20},{'ema':5,'dora':10,'alice':2
pd.DataFrame(dic,index=['a','b','c'])
```

Out[11]:

	alex	joe	ema	dora	alice
a	1.0	2.0	NaN	NaN	NaN
b	NaN	NaN	5.0	10.0	20.0
c	NaN	NaN	5.0	10.0	20.0

DataFrame operations

In [12]:
```
d3
```

Out[12]:

	a	b	c
x	2	3	4
y	4	5	6
z	1	2	3

In [13]:
```
print(d3['a'])
print(d3['b'])
```

```
x    2
y    4
z    1
Name: a, dtype: int64
x    3
y    5
z    2
Name: b, dtype: int64
```

In [14]:
```
d3['d']=d3['a']*d3['b']
d3['e']=d3['a']+d3['d']
d3
```

Out[14]:

	a	b	c	d	e
x	2	3	4	6	8
y	4	5	6	20	24
z	1	2	3	2	3

In [15]:
```
pop = d3.pop('b')
d3
```

Out[15]:

	a	c	d	e
x	2	4	6	8
y	4	6	20	24
z	1	3	2	3

In [16]:
```
d3
```

Out[16]:

	a	c	d	e
x	2	4	6	8
y	4	6	20	24
z	1	3	2	3

In [17]:
```
del d3['d']
```

In [18]:
```
d3
```

Out[18]:

	a	c	e
x	2	4	8
y	4	6	24
z	1	3	3

In [19]:
```
d3.pop('a')
d3
```

Out[19]:

	c	e
x	4	8
y	6	24
z	3	3

In [20]:
```
d3.insert(1,'new1',d3['c']) #(loc,col,data)
d3
```

Out[20]:

	c	new1	e
x	4	4	8
y	6	6	24
z	3	3	3

In [21]:
```
d={'c':1,'new1':33,'e':22}
df11=d3.append(d,ignore_index=True)
df11
```

Out[21]:

	c	new1	e
0	4	4	8
1	6	6	24
2	3	3	3
3	1	33	22

In [24]:
```
import numpy as np
d4 = pd.DataFrame({'abc':np.random.randint(2,6,size=(10)),'bcd':np.random.randint(4,1
d4
```

Out[24]:

	abc	bcd	cde
0	2	5	5
1	3	6	4
2	4	6	8
3	2	5	8
4	5	5	6
5	4	4	3
6	5	8	7
7	2	4	4
8	5	5	7
9	3	6	3

In [25]:
```
d4.head()
```

Out[25]:

	abc	bcd	cde
0	2	5	5
1	3	6	4
2	4	6	8
3	2	5	8
4	5	5	6

In [26]:
```
d4.tail()
```

Out[26]:

	abc	bcd	cde
5	4	4	3
6	5	8	7
7	2	4	4
8	5	5	7
9	3	6	3

In [27]:
```
'''
Info About the Data
The DataFrames object has a method called info(),that gives you more information about
d4.info()
d4.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   abc     10 non-null     int32
 1   bcd     10 non-null     int32
 2   cde     10 non-null     int32
dtypes: int32(3)
memory usage: 248.0 bytes
```

Out[27]:

	abc	bcd	cde
count	10.000000	10.000000	10.00000
mean	3.500000	5.400000	5.50000
std	1.269296	1.173788	1.95789
min	2.000000	4.000000	3.00000
25%	2.250000	5.000000	4.00000
50%	3.500000	5.000000	5.50000
75%	4.750000	6.000000	7.00000
max	5.000000	8.000000	8.00000

In [28]:
```
d4
```

Out[28]:

	abc	bcd	cde
0	2	5	5
1	3	6	4
2	4	6	8
3	2	5	8
4	5	5	6
5	4	4	3
6	5	8	7
7	2	4	4
8	5	5	7
9	3	6	3

In [29]:
```
d4.loc[9,'cde'] #loc[row name,colname]
```

Out[29]:
```
3
```

In [30]:
```
d4
```

Out[30]:

	abc	bcd	cde
0	2	5	5
1	3	6	4
2	4	6	8
3	2	5	8
4	5	5	6
5	4	4	3
6	5	8	7
7	2	4	4
8	5	5	7
9	3	6	3

In [31]:
```
d4.loc[4:9,['abc','cde']]
```

Out[31]:

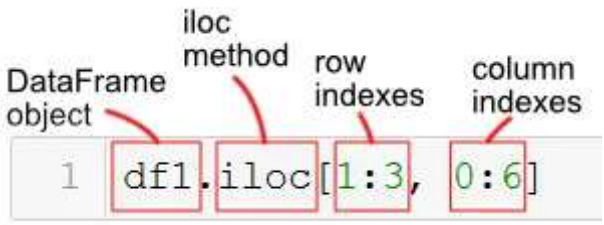
	abc	cde
4	5	6
5	4	3
6	5	7
7	2	4
8	5	7
9	3	3

In [32]:
```
d4.loc[[3,4,7],['abc','cde']]
```

Out[32]:

	abc	cde
3	2	8
4	5	6
7	2	4

In [ ]:
```
d4
```



In [33]:
```
d4.iloc[9,2]  #[row index,column index]
```

Out[33]:
```
3
```

In [34]:
```
d4.iloc[2:7,[0,2]]
```

Out[34]:

	abc	cde
2	4	8
3	2	8
4	5	6
5	4	3
6	5	7

In [35]:
```
d4
```

Out[35]:

	abc	bcd	cde
0	2	5	5
1	3	6	4
2	4	6	8
3	2	5	8
4	5	5	6
5	4	4	3
6	5	8	7
7	2	4	4
8	5	5	7
9	3	6	3

In [36]:
```
d4.abc
```

Out[36]:
```
0    2
1    3
2    4
3    2
4    5
5    4
6    5
7    2
8    5
9    3
Name: abc, dtype: int32
```

In [37]:
```
d4.abc.values # data frame . column .values
```

Out[37]:
```
array([2, 3, 4, 2, 5, 4, 5, 2, 5, 3])
```

In [38]:
```
d4['sub'] = d4.abc.values - d4.cde.values - d4.bcd.values
d4
```

Out[38]:

	abc	bcd	cde	sub
0	2	5	5	-8
1	3	6	4	-7
2	4	6	8	-10
3	2	5	8	-11
4	5	5	6	-6
5	4	4	3	-3
6	5	8	7	-10
7	2	4	4	-6
8	5	5	7	-7
9	3	6	3	-6

In [5]:
```
import pandas as pd
a = [['rk',102,15000],['rama',103,20000],['krishna',104,25000]]
dfl = pd.DataFrame(a,columns = ['name','id','salary'])
dfl
```

Out[5]:

	name	id	salary
0	rk	102	15000
1	rama	103	20000
2	krishna	104	25000

In [40]:
```
y=dfl[dfl.salary>=20000]
print(y)
y[['id','salary']]
```

```
      name   id  salary
1     rama  103   20000
2  krishna  104   25000
```

Out[40]:

	id	salary
1	103	20000
2	104	25000

In [41]:
```
dfl.append({'name' : 'sajan','id' : 105,'salary' : 30000},ignore_index=True)
```

Out[41]:

	name	id	salary
0	rk	102	15000
1	rama	103	20000
2	krishna	104	25000
3	sajan	105	30000

In [42]:
```
dfl = dfl.append({'name' : np.nan,'id' : 105,'salary' : 30000},ignore_index = True)
dfl
```

Out[42]:

	name	id	salary
0	rk	102.0	15000.0
1	rama	103.0	20000.0
2	krishna	104.0	25000.0
3	NaN	105.0	30000.0

In [43]:
```
dfl.isnull()
```

Out[43]:

	name	id	salary
0	False	False	False
1	False	False	False
2	False	False	False
3	True	False	False

In [44]:
```
print(dfl.isnull().sum())
```

```
name      1
id        0
salary    0
dtype: int64
```

In [45]:
```
dfl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   name    3 non-null      object 
 1   id      4 non-null      float64
 2   salary  4 non-null      float64
dtypes: float64(2), object(1)
memory usage: 224.0+ bytes
```

In [46]:
```
dfl.dropna()
```

Out[46]:

	name	id	salary
0	rk	102.0	15000.0
1	rama	103.0	20000.0
2	krishna	104.0	25000.0

In [47]:
```
dfl
```

Out[47]:

	name	id	salary
0	rk	102.0	15000.0
1	rama	103.0	20000.0
2	krishna	104.0	25000.0
3	NaN	105.0	30000.0

In [48]:
```
dfl.fillna(value = 'abc')
```

Out[48]:

	name	id	salary
0	rk	102.0	15000.0
1	rama	103.0	20000.0
2	krishna	104.0	25000.0
3	abc	105.0	30000.0

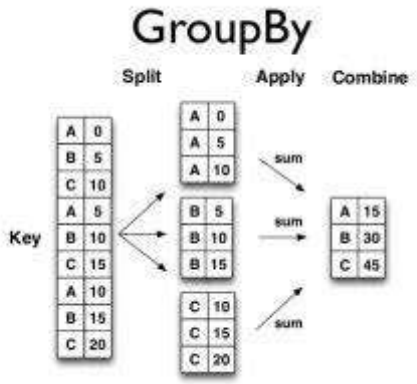
In [ ]:
```
dfl
```

In [49]:
```
dfl.drop([0],axis=0,inplace=True)
dfl
```

Out[49]:

	name	id	salary
1	rama	103.0	20000.0
2	krishna	104.0	25000.0
3	NaN	105.0	30000.0

Groupby



In [50]:
```
df = pd.DataFrame({'Animal' : ['Falcon', 'Falcon','Parrot', 'Parrot'],'Max Speed' : [
df
```

Out[50]:

	Animal	Max Speed	wind Speed
0	Falcon	380.0	380.0
1	Falcon	370.0	370.0
2	Parrot	24.0	24.0
3	Parrot	26.0	26.0

In [51]:
```
df.groupby(['Animal']).mean()
```

Out[51]:

Animal	Max Speed	wind Speed
Falcon	375.0	375.0
Parrot	25.0	25.0

In [52]:
```
import pandas as pd
df_csv = pd.read_csv('data.csv')
df_csv
```

Out[52]:

	Unnamed: 0	COUNTRY	POP	AREA	GDP	CONT	IND_DAY
0	CHN	China	1398.72	9596.96	12234.78	Asia	NaN
1	IND	India	1351.16	3287.26	2575.67	Asia	1947-08-15
2	USA	US	329.74	9833.52	19485.39	N.America	1776-07-04
3	IDN	Indonesia	268.07	1910.93	1015.54	Asia	1945-08-17
4	BRA	Brazil	210.32	8515.77	2055.51	S.America	1822-09-07
5	PAK	Pakistan	205.71	881.91	302.14	Asia	1947-08-14
6	NGA	Nigeria	200.96	923.77	375.77	Africa	1960-10-01
7	BGD	Bangladesh	167.09	147.57	245.63	Asia	1971-03-26
8	RUS	Russia	146.79	17098.25	1530.75	NaN	1992-06-12
9	MEX	Mexico	126.58	1964.38	1158.23	N.America	1810-09-16
10	JPN	Japan	126.22	377.97	4872.42	Asia	NaN
11	DEU	Germany	83.02	357.11	3693.20	Europe	NaN
12	FRA	France	67.02	640.68	2582.49	Europe	1789-07-14
13	GBR	UK	66.44	242.50	2631.23	Europe	NaN
14	ITA	Italy	60.36	301.34	1943.84	Europe	NaN
15	ARG	Argentina	44.94	2780.40	637.49	S.America	1816-07-09
16	DZA	Algeria	43.38	2381.74	167.56	Africa	1962-07-05
17	CAN	Canada	37.59	9984.67	1647.12	N.America	1867-07-01
18	AUS	Australia	25.47	7692.02	1408.68	Oceania	NaN
19	KAZ	Kazakhstan	18.53	2724.90	159.41	Asia	1991-12-16

In [53]:
```
print(df_csv.columns)
df_csv.isnull().sum()
```

```
Index(['Unnamed: 0', 'COUNTRY', 'POP', 'AREA', 'GDP', 'CONT', 'IND_DAY'], dtype='obje
ct')
```

Out[53]:
```
Unnamed: 0    0
COUNTRY       0
POP           0
AREA          0
GDP           0
CONT          1
IND_DAY       6
dtype: int64
```

In [54]:
```
df_csv['COUNTRY'].values
```

Out[54]:
```
array(['China', 'India', 'US', 'Indonesia', 'Brazil', 'Pakistan',
       'Nigeria', 'Bangladesh', 'Russia', 'Mexico', 'Japan', 'Germany',
       'France', 'UK', 'Italy', 'Argentina', 'Algeria', 'Canada',
       'Australia', 'Kazakhstan'], dtype=object)
```

In [55]:
```
df_csv['COUNTRY'].isnull().sum()
```

Out[55]:
```
0
```

In [56]:
```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(4,3),columns=['col1','col2','col3'])
print(df)
print('\n')
for key,value in df.iteritems():
   print(key,value)
   print('\n')
```

```
       col1      col2      col3
0 -0.123796  1.551366 -1.618906
1 -0.400873  0.346107 -1.033970
2 -1.735058 -1.172494  0.735198
3  0.299719 -0.789247  1.369722


col1 0   -0.123796
1   -0.400873
2   -1.735058
3    0.299719
Name: col1, dtype: float64


col2 0    1.551366
1    0.346107
2   -1.172494
3   -0.789247
Name: col2, dtype: float64


col3 0   -1.618906
1   -1.033970
2    0.735198
3    1.369722
Name: col3, dtype: float64
```

In [57]:
```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(3,3),columns = ['col1','col2','col3'])
print(df)
print('\n')
for row in df.iterrows():
   print(row)
   print('\n')
```

```
       col1      col2      col3
0 -0.667803  0.932184 -0.736992
1 -0.407077 -0.728226 -0.064054
2 -0.993813 -1.680033 -1.701452


(0, col1   -0.667803
col2    0.932184
col3   -0.736992
Name: 0, dtype: float64)


(1, col1   -0.407077
col2   -0.728226
col3   -0.064054
Name: 1, dtype: float64)


(2, col1   -0.993813
col2   -1.680033
col3   -1.701452
Name: 2, dtype: float64)
```

In [ ]:
```
#pd.Series?
```

In [58]:
```
# Transpose
df.T
```

Out[58]:

	0	1	2
col1	-0.667803	-0.407077	-0.993813
col2	0.932184	-0.728226	-1.680033
col3	-0.736992	-0.064054	-1.701452

Get Dummies

In [59]:
```
import pandas as pd

data = {'firstname': ['Arun', 'Jebu', 'Venkat', 'Rekha', 'Majid','Mohsin'],
        'lastname': ['Kumar', 'Jacob', 'Raghavan', 'Singh', 'Khan','Khan'],
        'employmenttype': ['Service', 'Business', 'Student', 'Service', 'Business','Bu
        'country' :['India','USA','USA','Sweden','Australia','Germany']}


df = pd.DataFrame(data, columns = ['firstname','lastname','employmenttype','country'])

print(df)
```

```
  firstname  lastname employmenttype    country
0      Arun     Kumar        Service      India
1      Jebu     Jacob       Business        USA
2    Venkat  Raghavan        Student        USA
3     Rekha     Singh        Service     Sweden
4     Majid      Khan       Business  Australia
5    Mohsin      Khan       Business    Germany
```

In [60]:
```
df1 = pd.get_dummies(df['employmenttype'])
df2 = pd.get_dummies(df['country'])
print(df1)
print(df2)
```

```
   Business  Service  Student
0         0        1        0
1         1        0        0
2         0        0        1
3         0        1        0
4         1        0        0
5         1        0        0
   Australia  Germany  India  Sweden  USA
0          0        0      1       0    0
1          0        0      0       0    1
2          0        0      0       0    1
3          0        0      0       1    0
4          1        0      0       0    0
5          0        1      0       0    0
```

In [ ]:
```
df
```

In [61]:
```
frames = [df,df1,df2]
result = pd.concat(frames,axis=1)
print(result)
result.drop(['employmenttype','country'],axis=1,inplace=True)
# inplace=true will perform any action permanently
result
```

```
  firstname  lastname employmenttype    country  Business  Service  Student  \
0      Arun     Kumar        Service      India         0        1        0
1      Jebu     Jacob       Business        USA         1        0        0
2    Venkat  Raghavan        Student        USA         0        0        1
3     Rekha     Singh        Service     Sweden         0        1        0
4     Majid      Khan       Business  Australia         1        0        0
5    Mohsin      Khan       Business    Germany         1        0        0

   Australia  Germany  India  Sweden  USA
0          0        0      1       0    0
1          0        0      0       0    1
2          0        0      0       0    1
3          0        0      0       1    0
4          1        0      0       0    0
5          0        1      0       0    0
```

Out[61]:

	firstname	lastname	Business	Service	Student	Australia	Germany	India	Sweden	USA
0	Arun	Kumar	0	1	0	0	0	1	0	0
1	Jebu	Jacob	1	0	0	0	0	0	0	1
2	Venkat	Raghavan	0	0	1	0	0	0	0	1
3	Rekha	Singh	0	1	0	0	0	0	1	0
4	Majid	Khan	1	0	0	1	0	0	0	0
5	Mohsin	Khan	1	0	0	0	1	0	0	0

concatination

In [62]:
```
import pandas as pd
india_weather = pd.DataFrame({
    "city": ["mumbai","delhi","banglore"],
    "temperature": [32,45,30],
    "humidity": [80, 60, 78]
})
india_weather
```

Out[62]:

	city	temperature	humidity
0	mumbai	32	80
1	delhi	45	60
2	banglore	30	78

In [63]:
```
us_weather = pd.DataFrame({
    "city": ["new york","chicago","orlando"],
    "temperature": [21,14,35],
    "humidity": [68, 65, 75]
})
us_weather
```

Out[63]:

	city	temperature	humidity
0	new york	21	68
1	chicago	14	65
2	orlando	35	75

In [64]:
```
df = pd.concat([india_weather, us_weather])
df
```

Out[64]:

	city	temperature	humidity
0	mumbai	32	80
1	delhi	45	60
2	banglore	30	78
0	new york	21	68
1	chicago	14	65
2	orlando	35	75

In [65]:
```
# ignore index
df = pd.concat([india_weather, us_weather], ignore_index=True)
df
```

Out[65]:

	city	temperature	humidity
0	mumbai	32	80
1	delhi	45	60
2	banglore	30	78
3	new york	21	68
4	chicago	14	65
5	orlando	35	75

In [66]:
```
# concatinate using keys
df = pd.concat([india_weather, us_weather], keys=["india", "us"])
df
```

Out[66]:

		city	temperature	humidity
india	**0**	mumbai	32	80
	1	delhi	45	60
	2	banglore	30	78
us	**0**	new york	21	68
	1	chicago	14	65
	2	orlando	35	75

In [67]:
```
'''
Finding Relationships
A great aspect of the Pandas module is the corr() method.

The corr() method calculates the relationship between each column in your data set.
'''
```

Out[67]:
```
'\nFinding Relationships\nA great aspect of the Pandas module is the corr() method.\n
\nThe corr() method calculates the relationship between each column in your data se
t.\n'
```

In [68]:
```
df=pd.read_csv('data_panda.csv')
df.corr()
```

Out[68]:

	Duration	Pulse	Maxpulse	Calories
Duration	1.000000	-0.155408	0.009403	0.922717
Pulse	-0.155408	1.000000	0.786535	0.025121
Maxpulse	0.009403	0.786535	1.000000	0.203813
Calories	0.922717	0.025121	0.203813	1.000000

In [ ]:
```
'''
Perfect Correlation:
We can see that "Duration" and "Duration" got the number 1.000000,
which makes sense, each column always has a perfect relationship with itself.

Good Correlation:
"Duration" and "Calories" got a 0.922721 correlation, which is a very good correlatior
and we can predict that the longer you work out, the more calories you burn, and the c
if you burned a lot of calories, you probably had a long work out.

Bad Correlation:
"Duration" and "Maxpulse" got a 0.009403 correlation, which is a very bad correlation,
meaning that we can not predict the max pulse by just looking at the duration of the w

'''
```

In [69]:
```
import pandas as pd
emp = {"Name": ["Parker", "Smith", "William", "Parker"],
"Age": [21, 32, 29, 21]}
info = pd.DataFrame(emp)
print(info)
print("--------------------")
info = info.drop_duplicates()
print(info)
```

```
      Name  Age
0   Parker   21
1    Smith   32
2  William   29
3   Parker   21
--------------------
      Name  Age
0   Parker   21
1    Smith   32
2  William   29
```

Apply

In [70]:
```
def calc_sum(kiran):
    return kiran+1
data = {
```

```
  "x": [50, 40, 30],
  "y": [300, 1112, 42]
}
df = pd.DataFrame(data)
print(df)
x = df.apply(calc_sum)
print(x)
```

```
    x     y
0  50   300
1  40  1112
2  30    42
    x     y
0  51   301
1  41  1113
2  31    43
```

In [71]:

```
# Categoricals
import pandas as pd
df=pd.read_html("https://github.com/datasciencedojo/datasets/blob/master/titanic.csv")
df1=df[0]
df1
```

Out[71]:

	Unnamed: 0	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	I
0	NaN	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	NaN	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	NaN	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	NaN	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	NaN	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0
...	...	...	...	...	...	...	...	...	...	...	
886	NaN	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0
887	NaN	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0
888	NaN	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4
889	NaN	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0
890	NaN	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7

891 rows × 13 columns

In [73]:
```
d=pd.Categorical(df1['Sex'])
print(d)
```

```
['male', 'female', 'female', 'female', 'male', ..., 'male', 'female', 'female', 'mal
e', 'male']
Length: 891
Categories (2, object): ['female', 'male']
```

In [74]:
```
import pandas as pd

df=pd.read_html('https://www.basketball-reference.com/teams/DEN/2023.html')
d=df[0]
d.head()
```

Out[74]:

	No.	Player	Pos	Ht	Wt	Birth Date	Unnamed: 6	Exp	College
0	11.0	Bruce Brown	SG	6-4	202	August 15, 1996	us	4	Miami (FL)
1	5.0	Kentavious Caldwell-Pope	SG	6-5	204	February 18, 1993	us	9	Georgia
2	0.0	Christian Braun	SG	6-7	218	April 17, 2001	us	R	Kansas
3	15.0	Nikola Jokić	C	6-11	284	February 19, 1995	rs	7	NaN
4	50.0	Aaron Gordon	PF	6-8	235	September 16, 1995	us	8	Arizona

In [75]:
```
df=d.set_index('Player')
df
```

Out[75]:

Player	No.	Pos	Ht	Wt	Birth Date	Unnamed: 6	Exp	College
Bruce Brown	11.0	SG	6-4	202	August 15, 1996	us	4	Miami (FL)
Kentavious Caldwell-Pope	5.0	SG	6-5	204	February 18, 1993	us	9	Georgia
Christian Braun	0.0	SG	6-7	218	April 17, 2001	us	R	Kansas
Nikola Jokić	15.0	C	6-11	284	February 19, 1995	rs	7	NaN
Aaron Gordon	50.0	PF	6-8	235	September 16, 1995	us	8	Arizona
Jamal Murray	27.0	PG	6-3	215	February 23, 1997	ca	5	Kentucky
Michael Porter Jr.	1.0	SF	6-10	218	June 29, 1998	us	3	Missouri
Vlatko Čančar	31.0	PF	6-8	236	April 10, 1997	si	3	NaN
Jeff Green	32.0	PF	6-8	235	August 28, 1986	us	14	Georgetown
Zeke Nnaji	22.0	PF	6-9	240	January 9, 2001	us	2	Arizona
DeAndre Jordan	6.0	C	6-11	265	July 21, 1988	us	14	Texas A&M
Ish Smith	14.0	PG	6-0	175	July 5, 1988	us	12	Wake Forest
Jack White (TW)	10.0	SF	6-7	225	August 5, 1997	au	R	Duke
Peyton Watson	8.0	SG	6-8	200	September 11, 2002	us	R	UCLA
Thomas Bryant	13.0	C	6-10	248	July 31, 1997	us	5	Indiana
Reggie Jackson	7.0	PG	6-2	208	April 16, 1990	it	11	Boston College
Collin Gillespie (TW)	NaN	PG	6-3	195	June 25, 1999	us	R	Villanova

In [ ]:

```
#explore sql joins
import pandas as pd
data1 = {
  "name": ["Sally", "Mary", "John"],
  "age": [50, 40, 30]
}

data2 = {
  "name": ["Sally", "Peter", "Micky"],
  "age": [77, 44, 22]
}

df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)
print(df1)
print(df2)
```

```
print("...........")
newdf = df1.merge(df2, how='right')
print(newdf)
```

In [ ]:
```
print(df1)
print(df2)
```

In [ ]:
```
df3=pd.merge(df1,df2,on="name",how="outer") # union
df3
```

In [ ]:
```
df3=pd.merge(df1,df2,on="name",how="left")
df3
```

In [ ]:
```
df3=pd.merge(df1,df2,on="name",how="right")
df3
```

In [ ]:
```
df3=pd.merge(df1,df2,on="name",how="inner") # intersection
df3
```

In [ ]:

In [ ]: