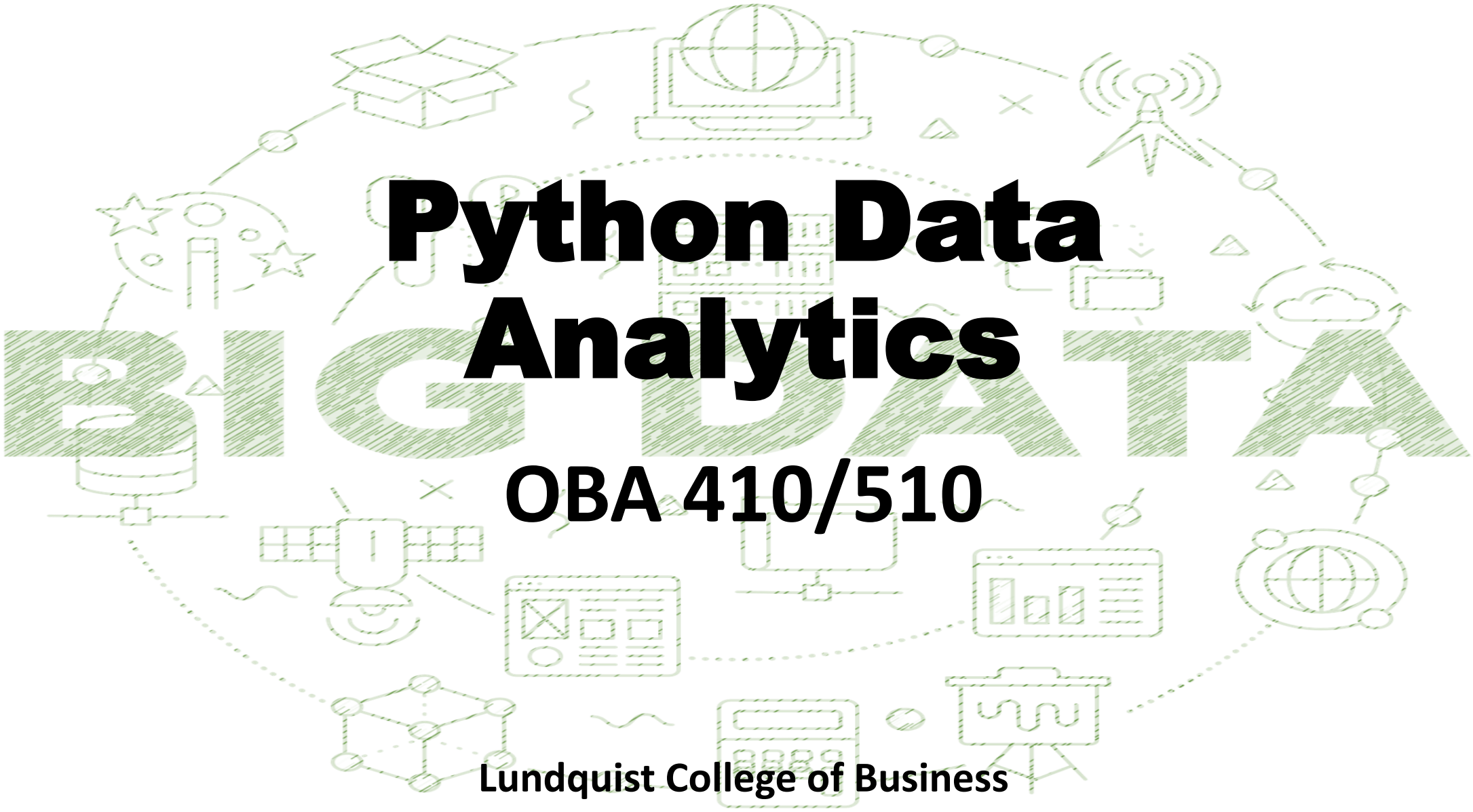


Python Data Analytics

OBA 410/510

Lundquist College of Business

Background image from www. givweb.com

Some of the slides and contents are adopted from Statistical Learning course by Dr. Trevor Hastie

Pima Indians Diabetes Data

- “This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.”
  - Pregnancies: Number of times pregnant
  - Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
  - BloodPressure: Diastolic blood pressure (mm Hg)
  - SkinThickness: Triceps skin fold thickness (mm)
  - Insulin: 2-Hour serum insulin (mu U/ml)
  - BMI: Body mass index (weight in kg/(height in m)^2)
  - DiabetesPedigreeFunction: Diabetes pedigree function
  - Age: Age (years)
  - Outcome: Class variable (0 or 1)

https://www.kaggle.com/uciml/pima-indians-diabetes-database/home
https://data.world/data-society/pima-indians-diabetes-database

Reading Datasets into a DataFrame

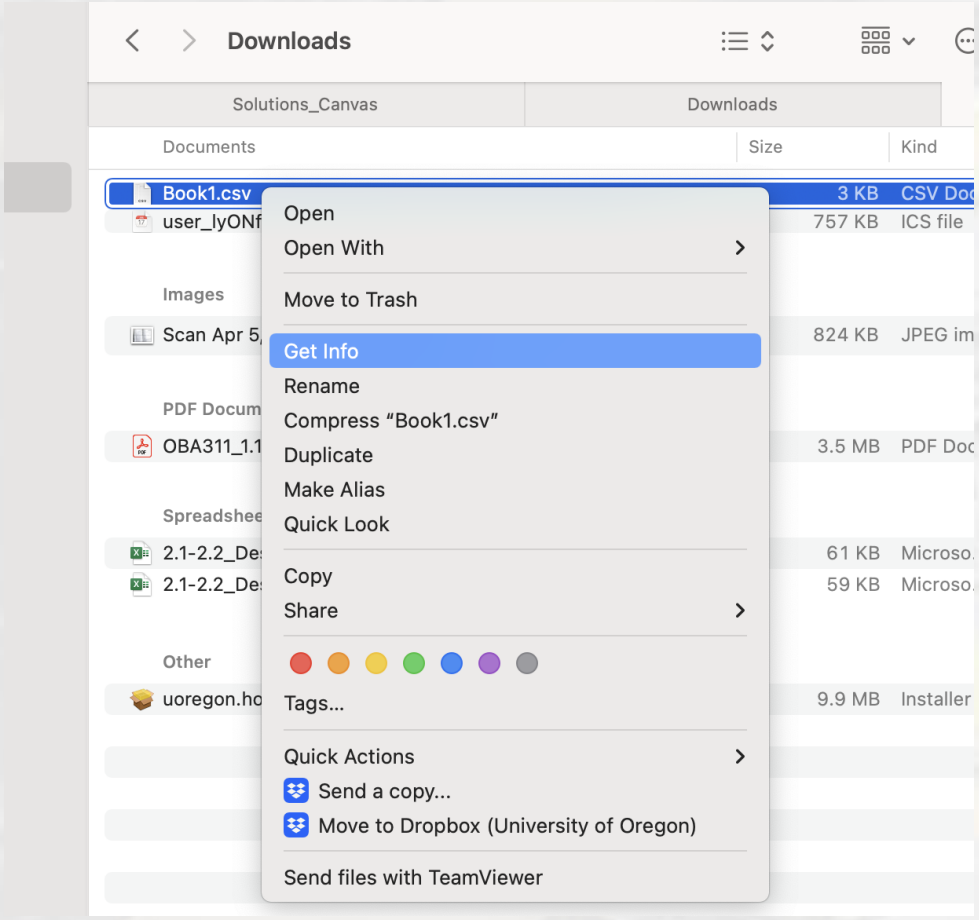
- pandas has built in support to read delaminate files such as CSV, Excel, etc.
  - CSV is a plain text format in which values are separated by commas (Comma Separated Values)
- When you specify a filename to pandas.read_csv, Python will look in your "current working directory".
  - Your working directory is the directory that you started your Jupiter notebook from
  - Finding your current working directory

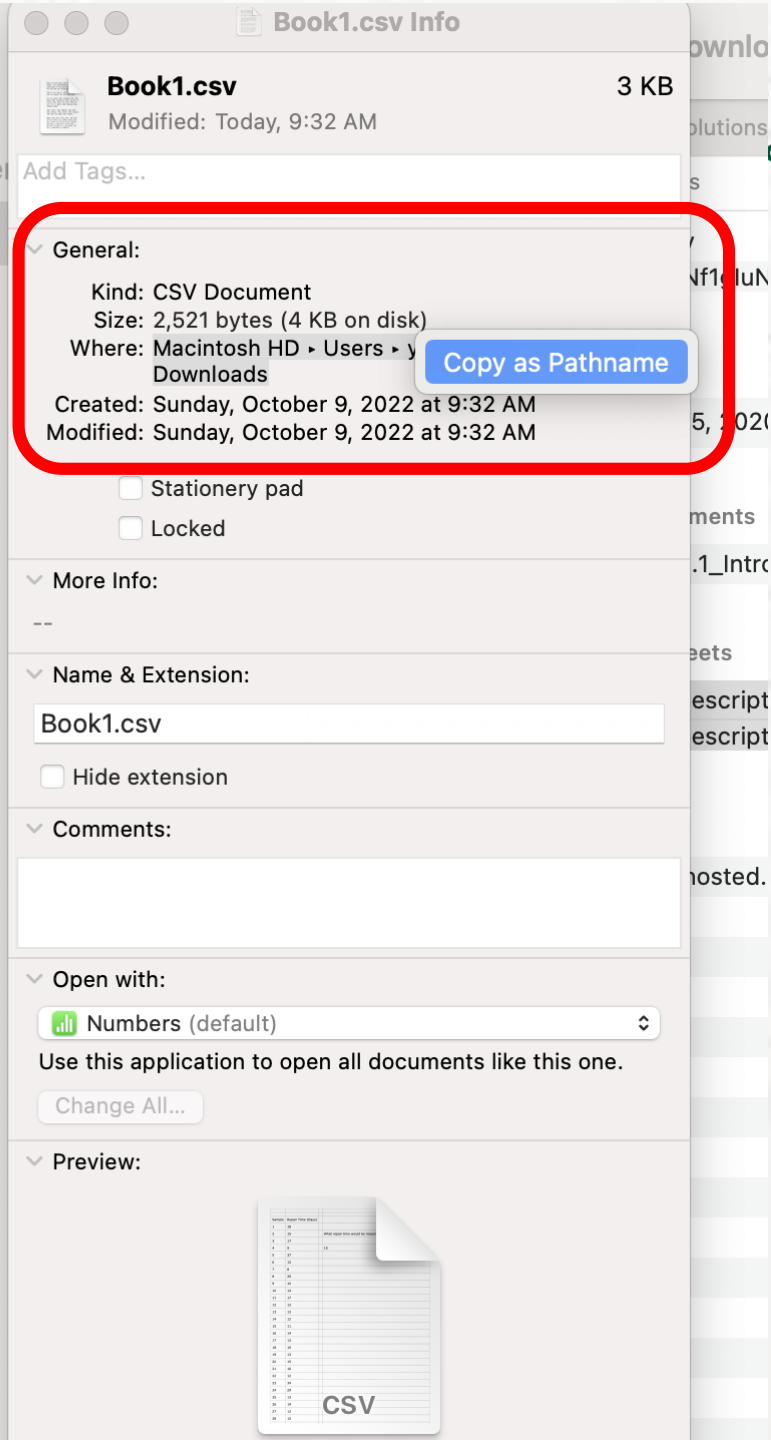
```
: # present working directory
  %pwd

: 'C:\\Users\\spiri'
```

  - If your file is in your working directory, simply having the name of the file is enough, otherwise, you need to provide the complete path

Getting the file path for MAC users

Right click on the file in your finder





Reading Datasets into a DataFrame

```
In [26]: df=pd.read_csv('diabetes.csv')
```

```
In [27]: df.head()
```

Out[27]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction		
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [30]: df2=pd.read_excel('diabetes.xlsx')
```

```
In [31]: df2.head()
```

Out[31]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [25]: x=pd.read_csv('D:/test/diabetes.csv')
         x.head()
```

Out[25]:

	Pregnancies	Glucose	BloodPressure	SkinThick
0	6	148	72	

DataFrames Components

- A DataFrame is composed of three different components:
  - Index,
  - Columns (labels),
  - values (or data)
- We can extract each of these components (and store them in variables)
  - index = df.index
  - columns = df.columns
  - values = df.values
    - …. (type: numpy.ndarray)

```
In [103]: df.index

Out[103]: RangeIndex(start=0, stop=768, step=1)

In [104]: df.columns

Out[104]: Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin'
                 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
                dtype='object')

In [105]: df.values

Out[105]: array([[  6.   , 148.   ,  72.   , ...,   0.627,  50.   ,   1.   ],
                 [  1.   ,  85.   ,  66.   , ...,   0.351,  31.   ,   0.   ],
                 [  8.   , 183.   ,  64.   , ...,   0.672,  32.   ,   1.   ],
                 ...,
                 [  5.   , 121.   ,  72.   , ...,   0.245,  30.   ,   0.   ],
                 [  1.   , 126.   ,  60.   , ...,   0.349,  47.   ,   1.   ],
                 [  1.   ,  93.   ,  70.   , ...,   0.315,  23.   ,   0.   ]])
```

Exploring DataFrames

- DataFrame.head(n=5) (*default 5)*
  - Return the first n rows
- DataFrame.tail(n)
  - Return the last n rows.
- DataFrame.info()
  - Print a concise summary of a DataFrame.
- DataFrame.shape
  - Return a tuple representing the dimensionality of the DataFrame.
- If you type the DataFrame name and execute it, pandas will auto detect the size of the displaying area and automatically hide some part of the data by replacing with ....
- If you want to see the whole data, you need to change the following settings:
  - pd.set_option('display.max_rows', 5000) (here I just set this to 5000!)
  - pd.set_option('display.max_columns', 500)
- If you want reset these options:

```
#resetting all display options
pd.reset_option('^display')
```

```
#resetting just max_rows diplay
pd.reset_option("display.max_rows")
```

UNIVERSITY OF OREGON
Lundquist College of Business

Exploring DataFrames

```
In [84]: df
```

24	11	143	94	33	146	36.6	0.254	51	1
25	10	125	70	26	115	31.1	0.205	41	1
26	7	147	76	0	0	39.4	0.257	43	1
27	1	97	66	15	140	23.2	0.487	22	0
28	13	145	82	19	110	22.2	0.245	57	0
29	5	117	92	0	0	34.1	0.337	38	0
...	...	...	...	...	...	...	...	...	...
739	1	102	74	0	0	39.5	0.293	42	1

```
In [85]: pd.set_option('display.max_rows', 5000)
         pd.set_option('display.max_columns', 500)
```

```
In [86]: df
```

28	13	145	82	19	110	22.2	0.245	57	0
29	5	117	92	0	0	34.1	0.337	38	0
30	5	109	75	26	0	36.0	0.546	60	0
31	3	158	76	36	245	31.6	0.851	28	1
32	3	88	58	11	54	24.8	0.267	22	0
33	6	92	92	0	0	19.9	0.188	28	0
34	10	122	78	31	0	27.6	0.512	45	0
35	4	103	60	33	192	24.0	0.966	33	0

Sorting a DataFrame

- We can sort a DataFrame using *.sort_values()* method
- This method can take multiple columns (they have to be passed to the function in a list)

```
In [414]: df.sort_values(['Pregnancies','Glucose'],ascending=False)
```

Out[414]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
159	17	163	72	41	114	40.9	0.817	47	1
88	15	136	70	32	110	37.1	0.153	43	1
455	14	175	62	30	0	33.6	0.212	38	1
298	14	100	78	25	184	36.6	0.412	46	1
691	13	158	114	0	0	42.3	0.257	44	1
744	13	153	88	37	140	40.6	1.174	39	0
323	13	152	90	33	29	26.8	0.731	43	1

Subsetting DataFrames

- we can subset DataFrames using:
- indexing operator ([ ]), used for column subsetting
- ***.loc*** used for rows or columns subsetting.
  - It can also simultaneously select subsets of rows and columns (it uses labels of rows and columns)
  - It is possible to 'slice' the rows of a DataFrame with .loc by using slice notation (**[:]**)
  - .loc includes the last value with slice notation
  - Selection by .loc can be a single label, a list of labels or a slice of labels
- ***.iloc***
  - The .iloc indexer is very similar to .loc but only uses integer locations to make its selections

Subsetting DataFrames ([ ])

- Using *indexing operator ([ ])*, we can select specifics columns from the data
- To select multiple columns, we need to pass a list of column names
- We actually can select rows with it, but it can be confusing and not used often

```
In [98]: df['Glucose']

Out[98]: 0      148
         1       85
         2      183
         3       89
         4      137
         5      116
         6       78
         7      115
         8      197
         9      125
         10     110
```

Series

```
In [106]: df[['Glucose']]

Out[106]:
```

	Glucose
0	148
1	85
2	183
3	89
4	137
5	116

DataFrame

```
In [99]: df[['Glucose','BloodPressure','BMI']]

Out[99]:
```

	Glucose	BloodPressure	BMI
0	148	72	33.6
1	85	66	26.6
2	183	64	23.3
3	89	66	28.1
4	137	40	43.1
5	116	74	25.6

DataFrame

Subsetting DataFrames (.loc)

UNIVERSITY OF OREGON
Lundquist College of Business

```
In [113]: df.loc[[1]]
```

DataFrame

Out[113]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	1	85	66	29	0	26.6	0.351	31	0

```
In [88]: df.loc[3:5]
```

Out[88]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0

```
In [89]: df.iloc[3:5]
```

Out[89]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [141]: df.loc[765:]
```

Out[141]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

```
In [114]: df.loc[1]
```

Out[114]:

```
Pregnancies                 1.000
Glucose                    85.000
BloodPressure              66.000
SkinThickness              29.000
Insulin                     0.000
BMI                        26.600
DiabetesPedigreeFunction    0.351
Age                        31.000
Outcome                     0.000
Name: 1, dtype: float64
```

Series

Subsetting DataFrames (.loc)

- Simultaneously select subsets of rows and columns
  - For example, glucose level of person number 5, you would supply two parameters to *loc*, one: the row index label and the other: the column name
  - df.loc[row_selection, column_selection]

```
In [142]: df.loc[4,'Glucose']
Out[142]: 137

In [143]: df.loc[[3,10,15,16],['Glucose','Age','BMI']]
Out[143]:
```

	Glucose	Age	BMI
3	89	21	28.1
10	110	30	37.6
15	100	32	30.0
16	118	31	45.8

```
In [146]: df.loc[3:6,'Glucose':'BMI']
Out[146]:
```

	Glucose	BloodPressure	SkinThickness	Insulin	BMI
3	89	66	23	94	28.1
4	137	40	35	168	43.1
5	116	74	0	0	25.6
6	78	50	32	88	31.0

Subsetting DataFrames (.iloc)

UNIVERSITY OF OREGON
Lundquist College of Business

```
In [148]: df.iloc[[0]]
```

Out[148]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1

```
In [150]: df.iloc[[1,2,6]]
```

Out[150]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
6	3	78	50	32	88	31.0	0.248	26	1

```
In [151]: df.iloc[3:6]
```

Out[151]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0

```
In [147]: df.iloc[0]
```

Out[147]:

```
Pregnancies                 6.000
Glucose                   148.000
BloodPressure              72.000
SkinThickness              35.000
Insulin                     0.000
BMI                        33.600
DiabetesPedigreeFunction    0.627
Age                        50.000
Outcome                     1.000
Name: 0, dtype: float64
```

```
In [152]: df.iloc[:3,0:4]
```

Out[152]:

	Pregnancies	Glucose	BloodPressure	SkinThickness
0	6	148	72	35
1	1	85	66	29
2	8	183	64	0

```
In [153]: df.iloc[[5,8,22],[0,3]]
```

Out[153]:

	Pregnancies	SkinThickness
5	5	0
8	2	45
22	7	0

Renaming indices and columns labels

[8]:
```
df5=df.iloc[:5].copy()
df5
```

t[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

[11]:
```
df5.rename(columns={'Pregnancies':'Num_of_Preg','BMI':'Body_Mass_Index','BloodPressure':'BP'},
           index={0:'p1',1:'p2',2:'p3',3:'p4',4:'p5'},inplace=True)
df5
```

[11]:

	Num_of_Preg	Glucose	BP	SkinThickness	Insulin	Body_Mass_Index	DiabetesPedigreeFunction	Age	Outcome
p1	6	148	72	35	0	33.6	0.627	50	1
p2	1	85	66	29	0	26.6	0.351	31	0
p3	8	183	64	0	0	23.3	0.672	32	1
p4	1	89	66	23	94	28.1	0.167	21	0
p5	0	137	40	35	168	43.1	2.288	33	1

Deleting data in DataFrames

- DataFrame.drop(labels=None, axis=0, index=None, columns=None, inplace=False)
  - labels : single label or list-like
    - Index or column labels to drop.
  - axis : {0 or ‘index’, 1 or ‘columns’}, default 0
    - Whether to drop labels from the index (0 or ‘index’) or columns (1 or ‘columns’).
  - index, columns : single label or list-like
    - Alternative to specifying axis (labels, axis=1 is equivalent to columns=labels).
  - inplace : bool, default False
    - If True, do operation inplace and return None.

```
2]: df5.drop(['p1','p4'])
```

2]:

	Num_of_Preg	Glucose	BP	SkinThickness	Insulin	Body_Mass_Index	DiabetesPedigreeFunction	Age	Outcome
p2	1	85	66	29	0	26.6	0.351	31	0
p3	8	183	64	0	0	23.3	0.672	32	1
p5	0	137	40	35	168	43.1	2.288	33	1

```
3]: df5.drop(index=['p1','p4'])
```

3]:

	Num_of_Preg	Glucose	BP	SkinThickness	Insulin	Body_Mass_Index	DiabetesPedigreeFunction	Age	Outcome
p2	1	85	66	29	0	26.6	0.351	31	0
p3	8	183	64	0	0	23.3	0.672	32	1
p5	0	137	40	35	168	43.1	2.288	33	1

```
: df5.drop(columns=['Glucose','Body_Mass_Index'])
```

:

	Num_of_Preg	BP	SkinThickness	Insulin	DiabetesPedigreeFunction	Age	Outcome
p1	6	72	35	0	0.627	50	1
p2	1	66	29	0	0.351	31	0
p3	8	64	0	0	0.672	32	1
p4	1	66	23	94	0.167	21	0
p5	0	40	35	168	2.288	33	1

```
df5.drop(index=['p2'],columns=['SkinThickness'],inplace=True)
df5
```

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	DiabetesPedigreeFunction	Age	Outcome
p1	6	148	72	0	33.6	0.627	50	1
p3	8	183	64	0	23.3	0.672	32	1
p4	1	89	66	94	28.1	0.167	21	0
p5	0	137	40	168	43.1	2.288	33	1

```
df5.drop('DiabetesPedigreeFunction',axis=1,inplace=True)
df5
```

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome
p1	6	148	72	0	33.6	50	1
p3	8	183	64	0	23.3	32	1
p4	1	89	66	94	28.1	21	0
p5	0	137	40	168	43.1	33	1

Adding a New Column to DataFrames

- It is as easy as assigning some value to it.

```
25]: df5['HbA1c']=None
     df5
```

25]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	DiabetesPedigreeFunction	Age	Outcome	HbA1c
p1	6	148	72	0	33.6	0.627	50	1	None
p3	8	183	64	0	23.3	0.672	32	1	None
p4	1	89	66	94	28.1	0.167	21	0	None
p5	0	137	40	168	43.1	2.288	33	1	None

```
26]: df5['HbA1c']=[34,41,52,61]
     df5
```

26]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	DiabetesPedigreeFunction	Age	Outcome	HbA1c
p1	6	148	72	0	33.6	0.627	50	1	34
p3	8	183	64	0	23.3	0.672	32	1	41
p4	1	89	66	94	28.1	0.167	21	0	52
p5	0	137	40	168	43.1	2.288	33	1	61

```
24]: df5['test']=(df5['Age']-10)/df5['HbA1c']
     df5
```

24]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome	HbA1c	test
p1	6	148	72	0	33.6	50	1	34	1.176471
p3	8	183	64	0	23.3	32	1	41	0.536585
p4	1	89	66	94	28.1	21	0	52	0.211538
p5	0	137	40	168	43.1	33	1	61	0.377049

Assigning Values

UNIVERSITY OF OREGON
Lundquist College of Business

```
]: df5.loc['p3','Insulin']=51
   df5
```

]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	DiabetesPedigreeFunction	Age	Outcome	HbA1c
p1	6	148	72	0	33.6	0.627	50	1	34
p3	8	183	64	51	23.3	0.672	32	1	41
p4	1	89	66	94	28.1	0.167	21	0	52
p5	0	137	40	168	43.1	2.288	33	1	61

```
]: df5.iloc[1,3]=70
   df5
```

]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	DiabetesPedigreeFunction	Age	Outcome	HbA1c
p1	6	148	72	0	33.6	0.627	50	1	34
p3	8	183	64	70	23.3	0.672	32	1	41
p4	1	89	66	94	28.1	0.167	21	0	52
p5	0	137	40	168	43.1	2.288	33	1	61

Adding new row

- The length of the list, have to be the same as the number of columns

```
]: df5.loc['p6']=None
   df5
```

]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome	HbA1c	test
p1	6.0	148.0	72.0	0.0	33.6	50.0	1.0	34.0	1.176471
p3	8.0	183.0	64.0	0.0	23.3	32.0	1.0	41.0	0.536585
p4	1.0	89.0	66.0	94.0	28.1	21.0	0.0	52.0	0.211538
p5	0.0	137.0	40.0	168.0	43.1	33.0	1.0	61.0	0.377049
p6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
]: df5.loc['p6']=[1,162,81,155,4,71,1,18,0]
   df5
```

]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome	HbA1c	test
p1	6.0	148.0	72.0	0.0	33.6	50.0	1.0	34.0	1.176471
p3	8.0	183.0	64.0	0.0	23.3	32.0	1.0	41.0	0.536585
p4	1.0	89.0	66.0	94.0	28.1	21.0	0.0	52.0	0.211538
p5	0.0	137.0	40.0	168.0	43.1	33.0	1.0	61.0	0.377049
p6	1.0	162.0	81.0	155.0	4.0	71.0	1.0	18.0	0.000000

Querying DataFrames

- selecting distinct records from a DataFrame:
- Unique values in a column:

For DataFrames:

```
df5[['Num_of_Preg']].drop_duplicates()
```

	Num_of_Preg
p1	6.0
p3	8.0
p4	1.0
p5	0.0

For Series:

```
df5['Num_of_Preg'].drop_duplicates()
```

```
p1    6.0
p3    8.0
p4    1.0
p5    0.0
Name: Num_of_Preg, dtype: float64
```

```
df5['Num_of_Preg'].unique()
```

```
array([6., 8., 1., 0.])
```

unique() works only for Series

```
df5.loc['p7']=df5.loc['p3']
df5
```

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome	HbA1c	test
p1	6.0	148.0	72.0	0.0	33.6	50.0	1.0	34.0	1.176471
p3	8.0	183.0	64.0	0.0	23.3	32.0	1.0	41.0	0.536585
p4	1.0	89.0	66.0	94.0	28.1	21.0	0.0	52.0	0.211538
p5	0.0	137.0	40.0	168.0	43.1	33.0	1.0	61.0	0.377049
p6	1.0	162.0	81.0	155.0	4.0	71.0	1.0	18.0	0.000000
p7	8.0	183.0	64.0	0.0	23.3	32.0	1.0	41.0	0.536585

```
df5.drop_duplicates()
```

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome	HbA1c	test
p1	6.0	148.0	72.0	0.0	33.6	50.0	1.0	34.0	1.176471
p3	8.0	183.0	64.0	0.0	23.3	32.0	1.0	41.0	0.536585
p4	1.0	89.0	66.0	94.0	28.1	21.0	0.0	52.0	0.211538
p5	0.0	137.0	40.0	168.0	43.1	33.0	1.0	61.0	0.377049
p6	1.0	162.0	81.0	155.0	4.0	71.0	1.0	18.0	0.000000

Querying DataFrames

UNIVERSITY OF OREGON
Lundquist College of Business

- Boolean mask:
  - We can use masking when we want to query or manipulate values in a DataFrame based on some criteria

```
df5
```

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Ag
p1	6.0	148.0	72.0	0.0	33.6	50.
p3	8.0	183.0	64.0	0.0	23.3	32.
p4	1.0	89.0	66.0	94.0	28.1	21.
p5	0.0	137.0	40.0	168.0	43.1	33.
p6	1.0	162.0	81.0	155.0	4.0	71.
p7	8.0	183.0	64.0	0.0	23.3	32.

```
gl_100=df5['Glucose']>100
gl_100
```

```
p1     True
p3     True
p4    False
p5     True
p6     True
p7     True
Name: Glucose, dtype: bool
```

```
df5[gl_100]
```

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	DiabetesPedigreeFunction	Age	Outcome	HbA1c	test
p1	6.0	148.0	72.0	0.0	33.6	0.627	50.0	1.0	34.0	1.470588
p3	8.0	183.0	64.0	70.0	23.3	0.672	32.0	1.0	41.0	0.780488
p5	0.0	137.0	40.0	168.0	43.1	2.288	33.0	1.0	61.0	0.540984

```
df5[df5['Glucose']>100]
```

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	DiabetesPedigreeFunction	Age	Outcome	HbA1c	test
p1	6.0	148.0	72.0	0.0	33.6	0.627	50.0	1.0	34.0	1.470588
p3	8.0	183.0	64.0	70.0	23.3	0.672	32.0	1.0	41.0	0.780488
p5	0.0	137.0	40.0	168.0	43.1	2.288	33.0	1.0	61.0	0.540984

Querying DataFrames

- More complicated queries using
  - Comparison operators
    - <, <=, >, >=, ==, !=
  - Logical
    - And → &
    - Or → |

```
8]: d=(df5['Glucose']>100) & (df5['Glucose']<150)
    d

8]: p1     True
    p3    False
    p4    False
    p5     True
    p6    False
    p7    False
    Name: Glucose, dtype: bool

0]: df5[d]
```

0]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome	Hb/
p1	6.0	148.0	72.0	0.0	33.6	50.0	1.0	3
p5	0.0	137.0	40.0	168.0	43.1	33.0	1.0	6

Querying DataFrames

UNIVERSITY OF OREGON
Lundquist College of Business

```
40]: df5[df5['Num_of_Preg']==1]
```

40]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome	HbA1c	test
p4	1.0	89.0	66.0	94.0	28.1				
p6	1.0	162.0	81.0	155.0	4.0				

```
41]: df5[(df5['Num_of_Preg']==1) & (df5['Body_Mass_Index']>27)]
```

41]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome	HbA1c	test
p4	1.0	89.0	66.0	94.0	28.1	21.0	0.0	52.0	0.211538

```
42]: a=(df5['Num_of_Preg']==1) & (df5['Body_Mass_Index']>27)
     a
```

```
42]: p1    False
     p3    False
     p4     True
     p5    False
     p6    False
     p7    False
     dtype: bool
```

```
44]: df5[a]
```

44]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome	HbA1c	test
p4	1.0	89.0	66.0	94.0	28.1	21.0	0.0	52.0	0.211538

Querying DataFrames

- parentheses are important!

```
54]: # patients (younger than 32 AND BP above 65)  OR  Glucose is 183
y_32=df5['Age']<32
bp_65=df5['BP']>65
gl_2=df5['Glucose']==183
con3=(y_32 & bp_65) | gl_2
df5[con3]
```

54]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome	HbA1c	test
p3	8.0	183.0	64.0	70.0	23.3	32.0	1.0	41.0	0.536585
p4	1.0	89.0	66.0	94.0	28.1	21.0	0.0	52.0	0.211538
p7	8.0	183.0	64.0	70.0	23.3	32.0	1.0	41.0	0.536585

```
58]: # patients younger than 32  AND  (BP above 65 OR Glucose is 183)
y_32=df5['Age']<32
bp_65=df5['BP']>65
gl_2=df5['Glucose']==183
con4=y_32 & (bp_65 | gl_2)
df5[con4]
```

58]:

	Num_of_Preg	Glucose	BP	Insulin	Body_Mass_Index	Age	Outcome	HbA1c	test
p4	1.0	89.0	66.0	94.0	28.1	21.0	0.0	52.0	0.211538

```
9]: df5[con4][['Num_of_Preg','Glucose']]
```

9]:

	Num_of_Preg	Glucose
p4	1.0	89.0

Summary Statistics

- *df.describe():* Generates descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution
- It ignores missing (NaN) values

```
In [259]: df.describe()
```

Out[259]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Missing Values

UNIVERSITY OF OREGON
Lundquist College of Business

- Real world datasets always have missing values
- To develop most predictive models, we need to fix missing values

```
In [379]: df_miss.shape
Out[379]: (48, 10)
```

```
In [382]: df_miss['Gender'].describe()
Out[382]: count          46
          unique          2
          top        Female
          freq           35
          Name: Gender, dtype: object
```

```
In [377]: df_miss=pd.read_csv('diabetes_w_missing.csv')
```

```
In [378]: df_miss
```

Out[378]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	Gender
0	6	148.0	72.0	35.0	0	33.6	0.627	50	1	Male
1	1	85.0	66.0	29.0	0	26.6	0.351	31	0	Male
2	8	183.0	64.0	0.0	0	23.3	0.672	32	1	Female
3	1	89.0	66.0	23.0	94	NaN	0.167	21	0	NaN
4	0	137.0	40.0	35.0	168	43.1	2.288	33	1	Male
5	5	116.0	74.0	0.0	0	25.6	0.201	30	0	Female
6	3	NaN	50.0	32.0	88	31.0	0.248	26	1	NaN
7	10	115.0	0.0	0.0	0	35.3	0.134	29	0	Female

```
In [381]: df_miss.describe()
```

Out[381]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	48.000000	43.000000	47.000000	46.000000	48.000000	42.000000	48.000000	48.000000	48.000000
mean	5.333333	129.255814	70.276596	19.195652	84.937500	31.564286	0.523354	37.708333	0.500000

Removing missing values

```
In [475]: # by default dropps rows with missing values
          df_miss.dropna()
```

Out[475]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	Gender
0	6	148.0	72.0	35.0	0	33.6	0.627	50	1	Male
1	1	85.0	66.0	29.0	0	26.6	0.351	31	0	Male
2	8	183.0	64.0	0.0	0	23.3	0.672	32	1	Female
4	0	137.0	40.0	35.0	168	43.1	2.288	33	1	Male
5	5	116.0	74.0	0.0	0	25.6	0.201	30	0	Female
7	10	115.0	0.0	0.0	0	35.3	0.134	29	0	Female
8	2	197.0	70.0	45.0	543	30.5	0.158	53	1	Female
11	10	168.0	74.0	0.0	0	38.0	0.537	34	1	Female
13	1	189.0	60.0	23.0	846	30.1	0.398	59	1	Female
14	5	166.0	72.0	19.0	175	25.8	0.587	51	1	Female
15	7	100.0	0.0	0.0	0	30.0	0.484	32	1	Female

```
In [477]: # dropps column with missing values
          df_miss.dropna(axis=1)
```

Out[477]:

	Pregnancies	Insulin	DiabetesPedigreeFunction	Age	Outcome
0	6	0	0.627	50	1
1	1	0	0.351	31	0
2	8	0	0.672	32	1
3	1	94	0.167	21	0
4	0	168	2.288	33	1
5	5	0	0.201	30	0

Imputing Missing Values

```
In [398]: Glucose_mean=np.mean(df_miss['Glucose'])
          SkinThickness_mean=np.mean(df_miss['SkinThickness'])
          Gender_mode=df_miss['Gender'].mode()
          imp_values={'Glucose':Glucose_mean,'SkinThickness':SkinThickness_mean,'BMI':20, 'Gender':Gender_mode[0]}
          df_miss.fillna(value=imp_values)
```

Out[398]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	Gender
0	6	148.000000	72.0	35.000000	0	33.6	0.627	50	1	Male
1	1	85.000000	66.0	29.000000	0	26.6	0.351	31	0	Male
2	8	183.000000	64.0	0.000000	0	23.3	0.672	32	1	Female
3	1	89.000000	66.0	23.000000	94	20.0	0.167	21	0	Female
4	0	137.000000	40.0	35.000000	168	43.1	2.288	33	1	Male
5	5	116.000000	74.0	0.000000	0	25.6	0.201	30	0	Female
6	3	129.255814	50.0	32.000000	88	31.0	0.248	26	1	Female
7	10	115.000000	0.0	0.000000	0	35.3	0.134	29	0	Female
8	2	197.000000	70.0	45.000000	543	30.5	0.158	53	1	Female
9	8	125.000000	96.0	19.195652	0	0.0	0.232	54	1	Female
10	4	110.000000	92.0	0.000000	0	20.0	0.191	30	0	Female

Imputing Missing Values

UNIVERSITY OF OREGON
Lundquist College of Business

```
In [9]: 1 df_miss.mean()

Out[9]: Pregnancies                 5.333333
        Glucose                   129.255814
        BloodPressure              70.276596
        SkinThickness              19.195652
        Insulin                    84.937500
        BMI                        31.564286
```

Imputing all numeric columns with their mean

```
In [8]: 1 df_miss.fillna(df_miss.mean())
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	Gende
0	6	148.000000	72.000000	35.000000	0	33.600000	0.627	50	1	Mal
1	1	85.000000	66.000000	29.000000	0	26.600000	0.351	31	0	Mal
2	8	183.000000	64.000000	0.000000	0	23.300000	0.672	32	1	Femal
3	1	89.000000	66.000000	23.000000	94	31.564286	0.167	21	0	Nal
4	0	137.000000	40.000000	35.000000	168	43.100000	2.288	33	1	Mal
5	5	116.000000	74.000000	0.000000	0	25.600000	0.201	30	0	Femal
6	3	129.255814	50.000000	32.000000	88	31.000000	0.248	26	1	Nal
7	10	115.000000	0.000000	0.000000	0	35.300000	0.134	29	0	Femal
8	2	197.000000	70.000000	45.000000	543	30.500000	0.158	53	1	Femal
9	8	125.000000	96.000000	19.195652	0	0.000000	0.232	54	1	Femal
10	4	110.000000	92.000000	0.000000	0	31.564286	0.191	30	0	Femal

Merging DataFrames

```
In [430]: Owners=pd.read_csv('Owners.csv')
          Pets=pd.read_csv('Pets.csv')
```

```
In [465]: Owners.head()
```

Out[465]:

	OwnerID	Name	Surname	StreetAddress	City	State	StateFull	ZipCode
0	6049	Debbie	Metivier	315 Goff Avenue	Grand Rapids	MI	Michigan	49503
1	2863	John	Sebastian	3221 Perry Street	Davison	MI	Michigan	48423
2	3518	Connie	Pauley	1539 Cunningham Court	Bloomfield Township	MI	Michigan	48302
3	3663	Lena	Haliburton	4217 Twin Oaks Drive	Traverse City	MI	Michigan	49684
4	1070	Jessica	Velazquez	3861 Woodbridge Lane	Southfield	MI	Michigan	48034

```
In [466]: Pets.head()
```

Out[466]:

	PetID	Name	Kind	Gender	Age	OwnerID
0	Z4-7776	Lack	Dog	male	3	4009
1	M0-2904	Simba	Cat	male	1	3086
2	R3-7551	Keller	Parrot	female	2	7908
3	T0-5705	Scout	Cat	female	5	5833
4	P0-1725	Lily	Dog	female	0	2419

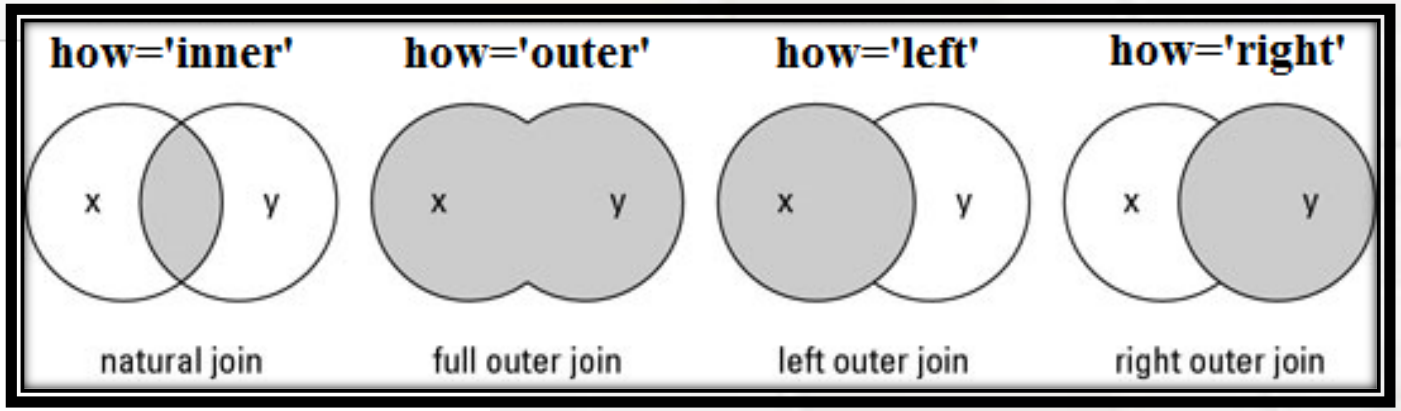


Photo from: http://www.datasciencemadesimple.com/join-merge-data-frames-pandas-python/

Inner join

```
In [467]: Owner_and_Pets=pd.merge(Owners,Pets,how='inner',on='OwnerID')
          Owner_and_Pets
```

Out[467]:

	OwnerID	Name_x	Surname	StreetAddress	City	State	StateFull	ZipCode	PetID	Name_y	Kind	Gender	Age
0	6049	Debbie	Metivier	315 Goff Avenue	Grand Rapids	MI	Michigan	49503	I6-9459	Biscuit	Dog	female	4
1	3518	Connie	Pauley	1539 Cunningham Court	Bloomfield Township	MI	Michigan	48302	N6-7350	Biscuit	Cat	female	8
2	3663	Lena	Haliburton	4217 Twin Oaks Drive	Traverse City	MI	Michigan	49684	U4-6674	Candy	Dog	female	1
3	1070	Jessica	Velazquez	3861 Woodbridge Lane	Southfield	MI	Michigan	48034	U8-6473	Cookie	Dog	female	3
4	2419	Luisa	Cuellar	1308 Shingleton Road	Kalamazoo	MI	Michigan	49007	P0-1725	Lily	Dog	female	0
5	6194	Karen	Torres	3941 Ritter Avenue	Center Line	MI	Michigan	48015	O6-3123	Lexie	Dog	female	2
6	5833	Mary	Hurtado	4865 Juniper Drive	Saint Charles	MI	Michigan	48655	T0-5705	Scout	Cat	female	5
7	9614	Carmen	Ingram	1056 Eagle Drive	Detroit	MI	Michigan	48219	X7-2632	Maripol	Parrot	female	7

Outer join

```
In [438]: pd.merge(Owners,Pets,how='outer',on='OwnerID')
```

Out[438]:

	OwnerID	Name_x	Surname	StreetAddress	City	State	StateFull	ZipCode	PetID	Name_y	Kind	Gender	Age
0	6049	Debbie	Metivier	315 Goff Avenue	Grand Rapids	MI	Michigan	49503.0	I6-9459	Biscuit	Dog	female	4.0
1	2863	John	Sebastian	3221 Perry Street	Davison	MI	Michigan	48423.0	NaN	NaN	NaN	NaN	NaN
2	3518	Connie	Pauley	1539 Cunningham Court	Bloomfield Township	MI	Michigan	48302.0	N6-7350	Biscuit	Cat	female	8.0
3	3663	Lena	Haliburton	4217 Twin Oaks Drive	Traverse City	MI	Michigan	49684.0	U4-6674	Candy	Dog	female	1.0
4	1070	Jessica	Velazquez	3861 Woodbridge Lane	Southfield	MI	Michigan	48034.0	U8-6473	Cookie	Dog	female	3.0
5	7101	Bessie	Yen	30 Cunningham Court	Rochester Hills	MI	Michigan	48306.0	NaN	NaN	NaN	NaN	NaN
6	2419	Luisa	Cuellar	1308 Shingleton Road	Kalamazoo	MI	Michigan	49007.0	P0-1725	Lily	Dog	female	0.0
7	6194	Karen	Torres	3941 Ritter Avenue	Center Line	MI	Michigan	48015.0	O6-3123	Lexie	Dog	female	2.0
8	5833	Mary	Hurtado	4865 Juniper Drive	Saint Charles	MI	Michigan	48655.0	T0-5705	Scout	Cat	female	5.0
9	9614	Carmen	Ingram	1056 Eagle Drive	Detroit	MI	Michigan	48219.0	X7-2632	Maripol	Parrot	female	7.0
10	7581	Florence	Nolen	3103 Howard Street	Grand Rapids	MI	Michigan	49503.0	NaN	NaN	NaN	NaN	NaN
11	2755	Anne	Hudson	4110 Howard Street	Grand Rapids	MI	Michigan	49503.0	NaN	NaN	NaN	NaN	NaN
12	9900	Marie	Floyd	314 Cunningham Court	Southfield	MI	Michigan	48075.0	NaN	NaN	NaN	NaN	NaN
13	8143	Jackie	Hatmaker	949 John Avenue	East Lansing	MI	Michigan	48823.0	NaN	NaN	NaN	NaN	NaN
14	4009	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Z4-7776	Lack	Dog	male	3.0
15	3086	NaN	NaN	NaN	NaN	NaN	NaN	NaN	M0-2904	Simba	Cat	male	1.0
16	7908	NaN	NaN	NaN	NaN	NaN	NaN	NaN	R3-7551	Keller	Parrot	female	2.0
17	4185	NaN	NaN	NaN	NaN	NaN	NaN	NaN	O5-2472	Bright	Dog	male	12.0
18	8133	NaN	NaN	NaN	NaN	NaN	NaN	NaN	O3-1895	Candy	Dog	female	3.0
19	7393	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N8-0553	Tiger	Dog	male	15.0
20	7340	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I5-4893	Cookie	Cat	female	3.0
21	9385	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Q8-0954	Lakshmi	Cat	female	7.0

Left join


```
In [484]: xx=pd.merge(Owners,Pets,how='left',on='OwnerID')
          xx
```

Out[484]:

	OwnerID	Name_x	Surname	StreetAddress	City	State	StateFull	ZipCode	PetID	Name_y	Kind	Gender	Age
0	6049	Debbie	Metivier	315 Goff Avenue	Grand Rapids	MI	Michigan	49503	I6-9459	Biscuit	Dog	female	4.0
1	2863	John	Sebastian	3221 Perry Street	Davison	MI	Michigan	48423	NaN	NaN	NaN	NaN	NaN
2	3518	Connie	Pauley	1539 Cunningham Court	Bloomfield Township	MI	Michigan	48302	N6-7350	Biscuit	Cat	female	8.0
3	3663	Lena	Haliburton	4217 Twin Oaks Drive	Traverse City	MI	Michigan	49684	U4-6674	Candy	Dog	female	1.0
4	1070	Jessica	Velazquez	3861 Woodbridge Lane	Southfield	MI	Michigan	48034	U8-6473	Cookie	Dog	female	3.0
5	7101	Bessie	Yen	30 Cunningham Court	Rochester Hills	MI	Michigan	48306	NaN	NaN	NaN	NaN	NaN

- If we want to keep only owners that we don’t their pets information

```
In [483]: xx[xx['PetID'].isnull()].dropna(axis=1)
```

Out[483]:

	OwnerID	Name_x	Surname	StreetAddress	City	State	StateFull	ZipCode
1	2863	John	Sebastian	3221 Perry Street	Davison	MI	Michigan	48423
5	7101	Bessie	Yen	30 Cunningham Court	Rochester Hills	MI	Michigan	48306
10	7581	Florence	Nolen	3103 Howard Street	Grand Rapids	MI	Michigan	49503
11	2755	Anne	Hudson	4110 Howard Street	Grand Rapids	MI	Michigan	49503
12	9900	Marie	Floyd	314 Cunningham Court	Southfield	MI	Michigan	48075
13	8143	Jackie	Hatmaker	949 John Avenue	East Lansing	MI	Michigan	48823

Saving Pandas DataFrames

- After processing and manipulating the data, we need to save it
- Saving to the working directory

```
In [222]: df2.to_excel('test.xlsx')

In [223]: df2.to_csv('test.csv')
```

- Saving to a specific path/directory

```
In [26]:  df2.to_csv('D:/test/diabetes22.csv')
```