TRANSFORMING DATA

Transformations

We'll consider four main transformations

Verb (tidyverse)	Functionality
mutate	Transform a column with some function
select	Select some columns in the data
filter	Keep only rows that meet some data criterion
arrange	Order the data frame by values of a column(s)

Tidy data: Journal of Statistical Software, Wickham (2014)

- Filter: subsetting or removing observations based on some condition.
- Transform: adding or modifying variables. These modifications can involve either a single variable (e.g., log-transformation), or multiple variables (e.g., computing density from weight and volume).
- Aggregate: collapsing multiple values into a single value (e.g., by summing or taking means).
- Sort: changing the order of observations.

These aren't explicity verbs in Python, but operationally they are similar to base R (non-tidyverse) manipulations

EXAMPLE:

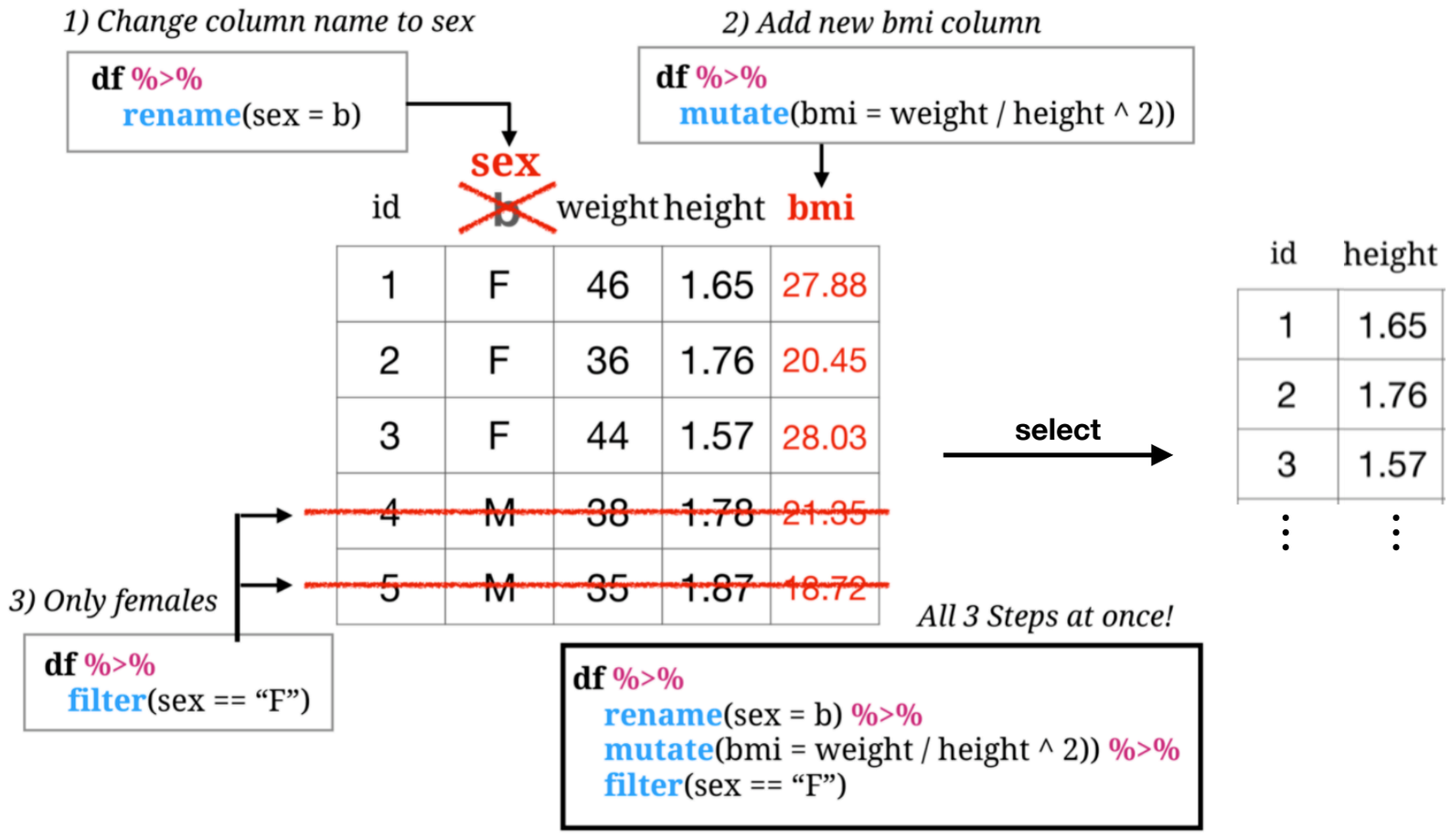


IMAGE SOURCE: https://therbootcamp.github.io/Erfurt_2018June/_sessions/D1S2_Wrangling/Wrangling_practical.html

MPG: PD example mpg.py

```
import pandas as pd

#-----------------------------
print("----READ----")
#-----------------------------
mpg = pd.read_csv('data/mpg.csv')
print(mpg.head(10))
```

original →

```
----READ----
   Unnamed: 0 manufacturer       model  displ  year  cyl       trans drv  cty  hwy fl    class
0           1          audi          a4    1.8  1999    4    auto(l5)   f   18   29  p  compact
1           2          audi          a4    1.8  1999    4  manual(m5)   f   21   29  p  compact
2           3          audi          a4    2.0  2008    4  manual(m6)   f   20   31  p  compact
3           4          audi          a4    2.0  2008    4    auto(av)   f   21   30  p  compact
4           5          audi          a4    2.8  1999    6    auto(l5)   f   16   26  p  compact
5           6          audi          a4    2.8  1999    6  manual(m5)   f   18   26  p  compact
6           7          audi          a4    3.1  2008    6    auto(av)   f   18   27  p  compact
7           8          audi  a4 quattro    1.8  1999    4  manual(m5)   4   18   26  p  compact
8           9          audi  a4 quattro    1.8  1999    4    auto(l5)   4   16   25  p  compact
9          10          audi  a4 quattro    2.0  2008    4  manual(m6)   4   20   28  p  compact
```

```
#-----------------------------
print("----MUTATE----")
#-----------------
# mile/gallon --> km/l
mpg['cty'] = mpg['cty'] * 1.609/3.7854
mpg['hwy'] = mpg['hwy'] * 1.609/3.7854
mpg['avg'] = (mpg.cty + mpg.hwy)/2
print(mpg.head())
```

→

```
----MUTATE----
   Unnamed: 0 manufacturer model  displ  year  cyl       trans drv       cty       hwy fl    class       avg
0           1         audi    a4    1.8  1999    4    auto(l5)   f  7.650975  12.326571  p  compact  9.988773
1           2         audi    a4    1.8  1999    4  manual(m5)   f  8.926137  12.326571  p  compact 10.626354
2           3         audi    a4    2.0  2008    4  manual(m6)   f  8.501083  13.176679  p  compact 10.838881
3           4         audi    a4    2.0  2008    4    auto(av)   f  8.926137  12.751625  p  compact 10.838881
4           5         audi    a4    2.8  1999    6    auto(l5)   f  6.800866  11.051408  p  compact  8.926137
```

```
#-----------------------------
print("----SELECT----")
#-----------------------------
print(mpg[['year','manufacturer','avg']])
```

→

```
----SELECT----
     year manufacturer        avg
0    1999         audi   9.988773
1    1999         audi  10.626354
2    2008         audi  10.838881
3    2008         audi  10.838881
4    1999         audi   8.926137
```

```
#-----------------------------
print("----FILTER----")
#-----------------------------
print(mpg[mpg.manufacturer=='audi'])
```

→

```
----FILTER----
   Unnamed: 0 manufacturer       model  displ  year  cyl       trans drv       cty       hwy fl    class       avg
0           1         audi          a4    1.8  1999    4    auto(l5)   f  7.650975  12.326571  p  compact  9.988773
1           2         audi          a4    1.8  1999    4  manual(m5)   f  8.926137  12.326571  p  compact 10.626354
2           3         audi          a4    2.0  2008    4  manual(m6)   f  8.501083  13.176679  p  compact 10.838881
3           4         audi          a4    2.0  2008    4    auto(av)   f  8.926137  12.751625  p  compact 10.838881
4           5         audi          a4    2.8  1999    6    auto(l5)   f  6.800866  11.051408  p  compact  8.926137
5           6         audi          a4    2.8  1999    6  manual(m5)   f  7.650975  11.051408  p  compact  9.351191
6           7         audi          a4    3.1  2008    6    auto(av)   f  7.650975  11.476462  p  compact  9.563718
7           8         audi  a4 quattro    1.8  1999    4  manual(m5)   4  7.650975  11.051408  p  compact  9.351191
```

```
#-----------------------------
print("----ARRANGE----")
#-----------------------------
print(mpg.sort_values(by=['avg'],
ascending=False))
```

→

```
----ARRANGE----
     Unnamed: 0 manufacturer            model  displ  year  cyl       trans drv       cty       hwy fl       class       avg
221         222   volkswagen       new beetle    1.9  1999    4  manual(m5)   f  14.876895  18.702383  d  subcompact 16.789639
212         213   volkswagen            jetta    1.9  1999    4  manual(m5)   f  14.026787  18.702383  d     compact 16.364585
222         223   volkswagen       new beetle    1.9  1999    4    auto(l4)   f  12.326571  17.427220  d  subcompact 14.876895
196         197       toyota          corolla    1.8  2008    4  manual(m5)   f  11.901516  15.727004  r     compact 13.814260
195         196       toyota          corolla    1.8  1999    4  manual(m5)   f  11.051408  14.876895  r     compact 12.964152
```

MPG: R example

mpg.r

```r
library(tidyverse)

#----------------------------
print("----READ----")
#----------------------------
mpg1 <- read_csv('data/mpg.csv')
head(mpg1)

#----------------------------
print("----MUTATE----")
#----------------------------
# OPTION-1: SEND MPG TO MUTATE
 mpg2 <- mutate(mpg1, cty = cty * 1.6/3.8,
         hwy = hwy * 1.6/3.8,
         avg = (cty + hwy)/2)
head(mpg2); #print(mpg2['avg'])

#----------------------------
print("----MUTATE----")
#----------------------------
#OPTION-2: PIPE MPG INTO MUTATE
mpg1 <- mpg1 %>% mutate(cty = cty * 1.6/3.8,
         hwy = hwy * 1.6/3.8,
         avg = (cty + hwy)/2)
head(mpg1)

#----------------------------
print("----SELECT----")
#----------------------------
mpg2 <- mpg1 %>% select(year, manufacturer, avg)
head(mpg2)

#----------------------------
print("----FILTER----")
#----------------------------
mpg2 <- mpg1 %>% filter(manufacturer=='audi')
head(mpg2)

#----------------------------
print("----ARRANGE----")
#----------------------------
mpg2 <-mpg1 %>% arrange(-year)
head(mpg2)

#----------------------------
print("----FILTER THEN ARRANGE----")
#----------------------------
#CHAIN TOGETHER
mpg2 <-mpg1 %>% filter(manufacturer=='audi') %>%
arrange(-year)
head(mpg2)
```

OUTPUT

```
# A tibble: 6 × 12
   ...1 manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
  <dbl> <chr>        <chr> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl> <chr> <chr>
1     1 audi         a4      1.8  1999     4 auto… f        18    29 p     comp…
2     2 audi         a4      1.8  1999     4 manu… f        21    29 p     comp…
3     3 audi         a4      2    2008     4 manu… f        20    31 p     comp…
4     4 audi         a4      2    2008     4 auto… f        21    30 p     comp…
5     5 audi         a4      2.8  1999     6 auto… f        16    26 p     comp…
6     6 audi         a4      2.8  1999     6 manu… f        18    26 p     comp…
[1] "----MUTATE----"
# A tibble: 6 × 13
   ...1 manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
  <dbl> <chr>        <chr> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl> <chr> <chr>
1     1 audi         a4      1.8  1999     4 auto… f      7.58  12.2 p     comp…
2     2 audi         a4      1.8  1999     4 manu… f      8.84  12.2 p     comp…
3     3 audi         a4      2    2008     4 manu… f      8.42  13.1 p     comp…
4     4 audi         a4      2    2008     4 auto… f      8.84  12.6 p     comp…
5     5 audi         a4      2.8  1999     6 auto… f      6.74  10.9 p     comp…
6     6 audi         a4      2.8  1999     6 manu… f      7.58  10.9 p     comp…
# … with 1 more variable: avg <dbl>
[1] "----MUTATE----"
# A tibble: 6 × 13
   ...1 manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
  <dbl> <chr>        <chr> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl> <chr> <chr>
1     1 audi         a4      1.8  1999     4 auto… f      7.58  12.2 p     comp…
2     2 audi         a4      1.8  1999     4 manu… f      8.84  12.2 p     comp…
3     3 audi         a4      2    2008     4 manu… f      8.42  13.1 p     comp…
4     4 audi         a4      2    2008     4 auto… f      8.84  12.6 p     comp…
5     5 audi         a4      2.8  1999     6 auto… f      6.74  10.9 p     comp…
6     6 audi         a4      2.8  1999     6 manu… f      7.58  10.9 p     comp…
# … with 1 more variable: avg <dbl>
[1] "----SELECT----"
# A tibble: 6 × 3
   year manufacturer   avg
  <dbl> <chr>        <dbl>
1  1999 audi          9.89
2  1999 audi         10.5
3  2008 audi         10.7
4  2008 audi         10.7
5  1999 audi          8.84
6  1999 audi          9.26
[1] "----FILTER----"
# A tibble: 6 × 13
   ...1 manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
  <dbl> <chr>        <chr> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl> <chr> <chr>
1     1 audi         a4      1.8  1999     4 auto… f      7.58  12.2 p     comp…
2     2 audi         a4      1.8  1999     4 manu… f      8.84  12.2 p     comp…
3     3 audi         a4      2    2008     4 manu… f      8.42  13.1 p     comp…
4     4 audi         a4      2    2008     4 auto… f      8.84  12.6 p     comp…
5     5 audi         a4      2.8  1999     6 auto… f      6.74  10.9 p     comp…
6     6 audi         a4      2.8  1999     6 manu… f      7.58  10.9 p     comp…
# … with 1 more variable: avg <dbl>
[1] "----ARRANGE----"
# A tibble: 6 × 13
   ...1 manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
  <dbl> <chr>        <chr> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl> <chr> <chr>
1     3 audi         a4      2    2008     4 manu… f      8.42  13.1 p     comp…
2     4 audi         a4      2    2008     4 auto… f      8.84  12.6 p     comp…
3     7 audi         a4      3.1  2008     6 auto… f      7.58  11.4 p     comp…
4    10 audi         a4 q…   2    2008     4 manu… 4      8.42  11.8 p     comp…
5    11 audi         a4 q…   2    2008     4 auto… 4      8     11.4 p     comp…
6    14 audi         a4 q…   3.1  2008     6 auto… 4      7.16  10.5 p     comp…
# … with 1 more variable: avg <dbl>
[1] "----FILTER THEN ARRANGE----"
# A tibble: 6 × 13
   ...1 manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
  <dbl> <chr>        <chr> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl> <chr> <chr>
1     3 audi         a4      2    2008     4 manu… f      8.42  13.1 p     comp…
2     4 audi         a4      2    2008     4 auto… f      8.84  12.6 p     comp…
3     7 audi         a4      3.1  2008     6 auto… f      7.58  11.4 p     comp…
4    10 audi         a4 q…   2    2008     4 manu… 4      8.42  11.8 p     comp…
5    11 audi         a4 q…   2    2008     4 auto… 4      8     11.4 p     comp…
6    14 audi         a4 q…   3.1  2008     6 auto… 4      7.16  10.5 p     comp…
# … with 1 more variable: avg <dbl>
```

Changing data types

Datatypes:

R	Python
numeric	float
integer	int
character	str
factor	category

Both R and Python have the same data types.
The Categorical type below is unique to **pandas**

Categoricals are a pandas data type corresponding to categorical variables in statistics. A categorical variable takes on a limited, and usually fixed, number of possible values

Examples: gender, social class, blood type, country affiliation

R:

```
#----------------------------
#CONVERT TO FACTOR
#----------------------------
mpg2['model']=as.factor(mpg2$model)
head(mpg2)
```

```
   ...1 manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
  <dbl> <chr>        <chr> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl> <chr> <chr>
1     3 audi         a4      2    2008     4 manu… f      8.42  13.1 p     comp…
2     4 audi         a4      2    2008     4 auto… f      8.84  12.6 p     comp…
3     7 audi         a4      3.1  2008     6 auto… f      7.58  11.4 p     comp…
# … with 1 more variable: avg <dbl>
```

```
# A tibble: 3 × 13
   ...1 manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
  <dbl> <chr>        <fct> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl> <chr> <chr>
1     3 audi         a4      2    2008     4 manu… f      8.42  13.1 p     comp…
2     4 audi         a4      2    2008     4 auto… f      8.84  12.6 p     comp…
3     7 audi         a4      3.1  2008     6 auto… f      7.58  11.4 p     comp…
```

Python:

```
#----------------------------
print("----READ----")
#----------------------------
mpg = pd.read_csv('data/mpg.csv')
print(mpg.head(10))

#----------------------------
print("----CONVERT----")
#----------------------------
print(mpg.dtypes)
mpg['manufacturer']=mpg.manufacturer.astype('category')
mpg['model']=mpg.model.astype('string')
print('--------')
print(mpg.dtypes)
```

```
----CONVERT----
Unnamed: 0          int64
manufacturer       object
model              object
displ             float64
year                int64
.
.
--------
Unnamed: 0          int64
manufacturer    category
model             string
displ             float64
year                int64
.
.
```

JOINING DATASETS

- Quite often, data on individuals lie in different tables
  - Clinical, demographic and bioinformatic data
  - Drug, procedure, and payment data (think Medicare)
  - Personal health data across different healthcare entities

Pandas Concatenating

The simplest case is when we just need to add more data to existing data

Concatinating Series:

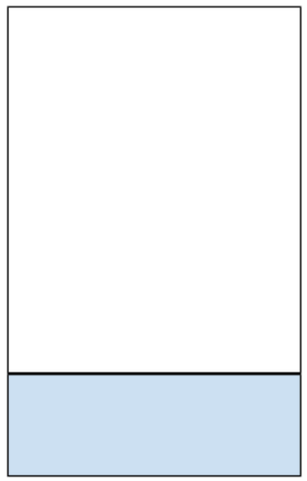
```
import pandas as pd

#CONCATINATING SERIES
s1 = pd.Series(['a', 'b'])
s2 = pd.Series(['c', 'd'])
print(s1)
print("----CONCAT SERIES---")
print(pd.concat([s1, s2],axis=0))
print(pd.concat([s1, s2],axis=1))
```

```
----CONCAT SERIES---
0    a
1    b
dtype: object
0    a
1    b
0    c
1    d
dtype: object
   0  1
0  a  c
1  b  d
```

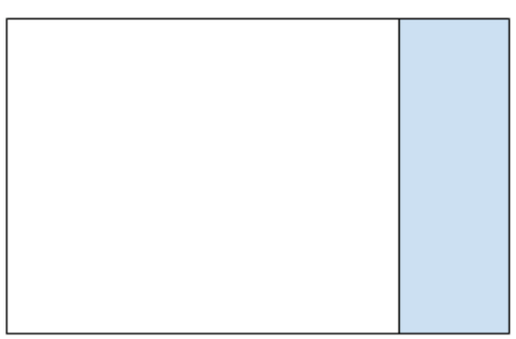
Concatinating dataframes:

Pandas: pd.concat(x,y, axis=0)



Add rows

Pandas: pd.concat(x,y, axis=1)



Add columns

```
import pandas as pd

#CONCATINATING DATA FRAMES
print("----CONCAT DATAFRAMES---")
df1 = pd.DataFrame([['a', 1], ['b', 2]], columns=['letter', 'number'])
df2 = pd.DataFrame([['c', 3], ['d', 4]], columns=['letter', 'number'])
print("INPUTS:")
print(df1)
print(df2)
print("OUTPUTS:")
print(pd.concat([df1, df2],axis=0))
print(pd.concat([df1, df2],axis=1))
```

```
----CONCAT DATAFRAMES---
INPUTS:
  letter  number
0      a       1
1      b       2
  letter  number
0      c       3
1      d       4
```

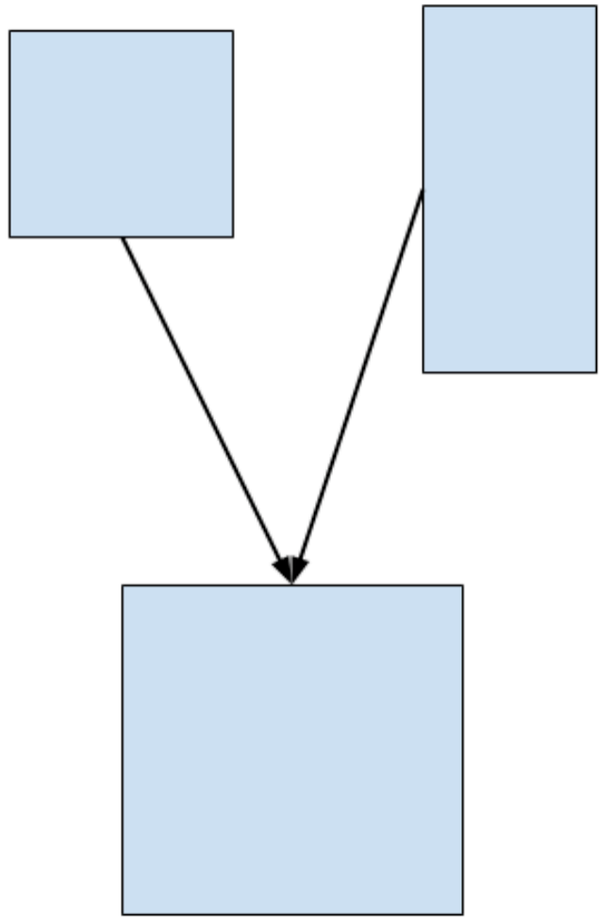
```
OUTPUTS:
  letter  number
0      a       1
1      b       2
0      c       3
1      d       4
  letter  number letter  number
0      a       1      c       3
1      b       2      d       4
```

Joins

We will talk about more general ways of joining two datasets

We will assume:

1. We have two rectangular data sets (so data.frame or tibble)
2. **There is at least one variable (column) in common**, even if they have different names
   - Patient ID number
   - SSN (Social Security number)
   - Identifiable information

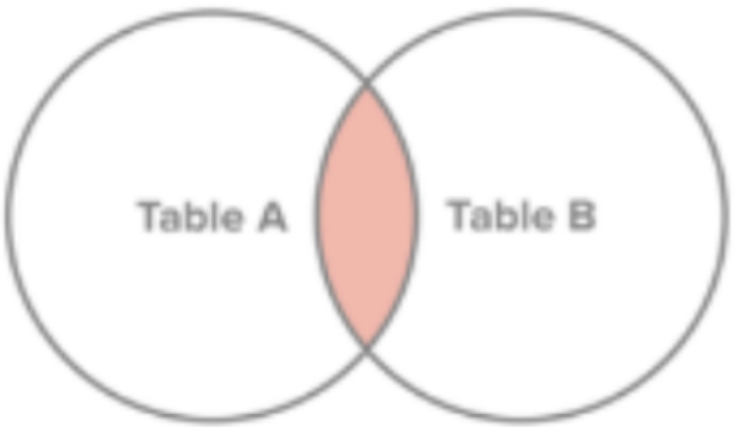


The **"join condition" are the common variables in the two datasets**, i.e. rows are selected if the values of the common variables in the left dataset matches the values of the common variables in the right dataset

Different types of joins

pandas	R	SQL	Description
left	left_join	left outer	keep all rows on left
right	right_join	right outer	keep all rows on right
outer	full_join	full outer	keep all rows from both
inner	inner_join	inner	keep only rows with common keys

Inner Joins

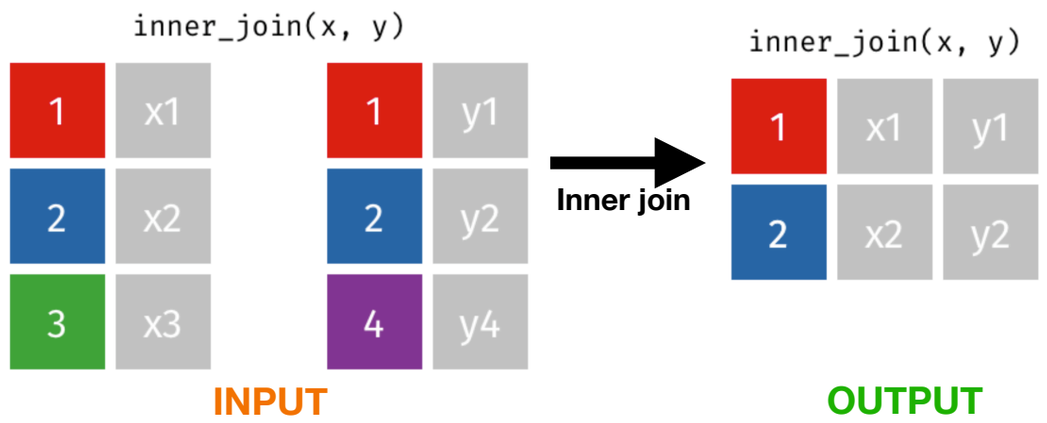


Select all records from Table A and Table B, where the join condition is met.

intersection of two sets A and B, denoted by $A \cap B$

$$A \cap B = \{x : x \in A \textbf{ and } x \in B\}.$$

$$\{1,2,3\} \cap \{2,3,4\} = \{2,3\} \qquad \{1,2,3\} \cap \{4,5,6\} = \emptyset$$



EXAMPLE:

```
print("----JOINING DATAFRAMES---")
df1 = pd.DataFrame([['x1', 'y1'], ['x2', 'y2'], ['x3', 'y3']], columns=['x', 'y'])
df2 = pd.DataFrame([['x1', 'y1','w1', 'z1']
                  , [‘x2', 'y2', 'w2', 'z2']
                  , [‘x4', 'y4', 'w4', 'z4']],columns=['x', 'y','w', 'z'])
print("INPUTS:")
print(df1,'\n')
print(df2)

print("------")
print('INNER: \n',df1.merge(df2, how='inner'))
```

→

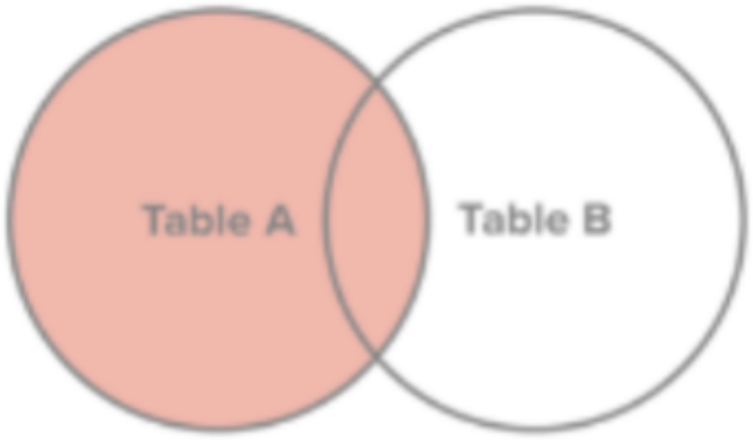
```
INPUTS:
    x   y
0  x1  y1
1  x2  y2
2  x3  y3

    x   y   w   z
0  x1  y1  w1  z1
1  x2  y2  w2  z2
2  x4  y4  w4  z4
```

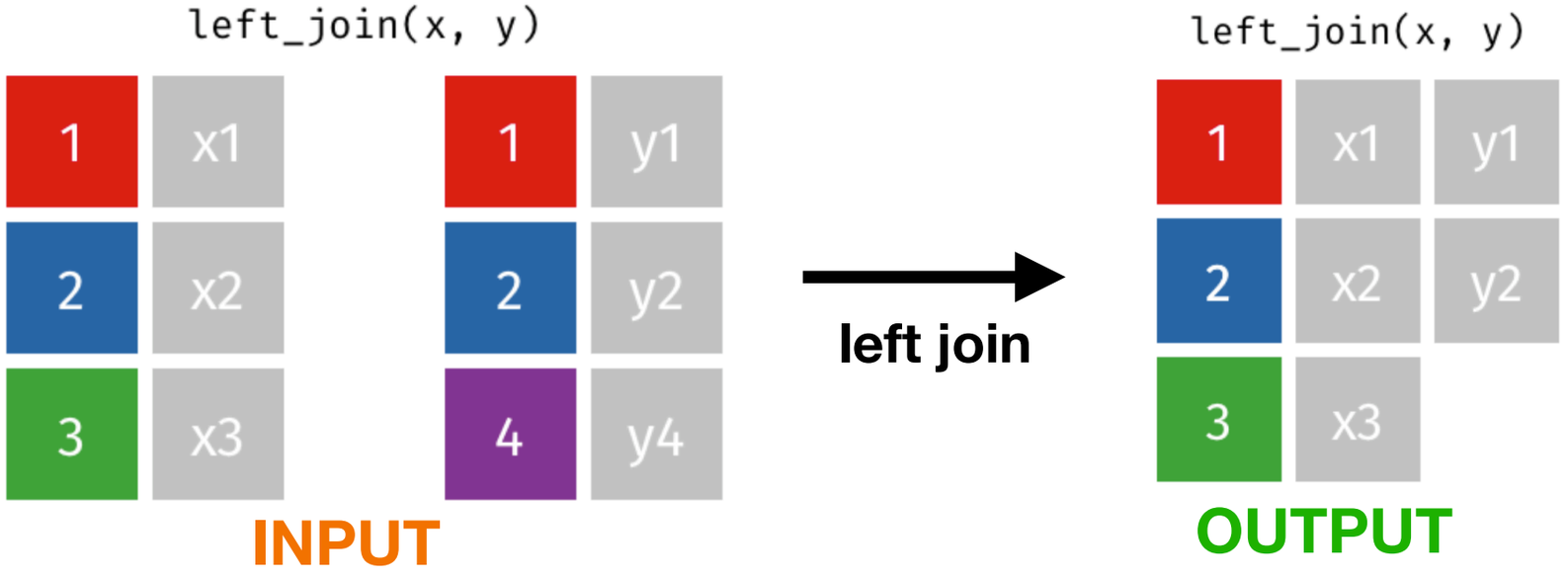
→

```
INNER JOIN:
    x   y   w   z
0  x1  y1  w1  z1
1  x2  y2  w2  z2
```

Left Joins



Select all records from Table A,
along with records from Table B
for which the join condition is met
(if at all).



EXAMPLE:

```
print("----JOINING DATAFRAMES---")
df1 = pd.DataFrame([['x1', 'y1'], ['x2', 'y2'], ['x3', 'y3']], columns=['x', 'y'])
df2 = pd.DataFrame([['x1', 'y1','w1', 'z1']
                  , [‘x2', 'y2', 'w2', 'z2']
                  , [‘x4', 'y4', 'w4', 'z4']],columns=['x', 'y','w', 'z'])
print("INPUTS:")
print(df1,'\n')
print(df2)

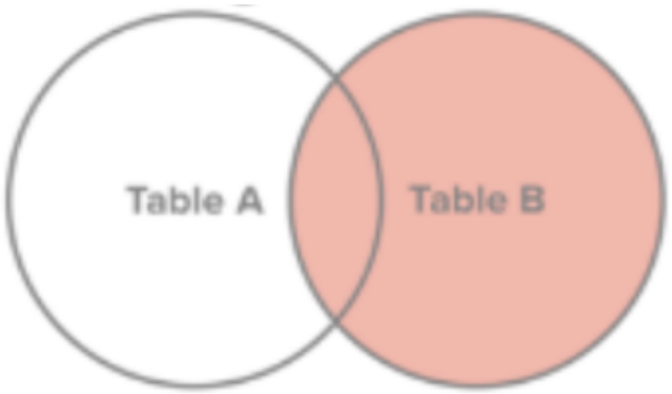
print("------")
print(‘LEFT JOIN:  \n',df1.merge(df2, how='left'))
```

```
INPUTS:
    x   y
0  x1  y1
1  x2  y2
2  x3  y3

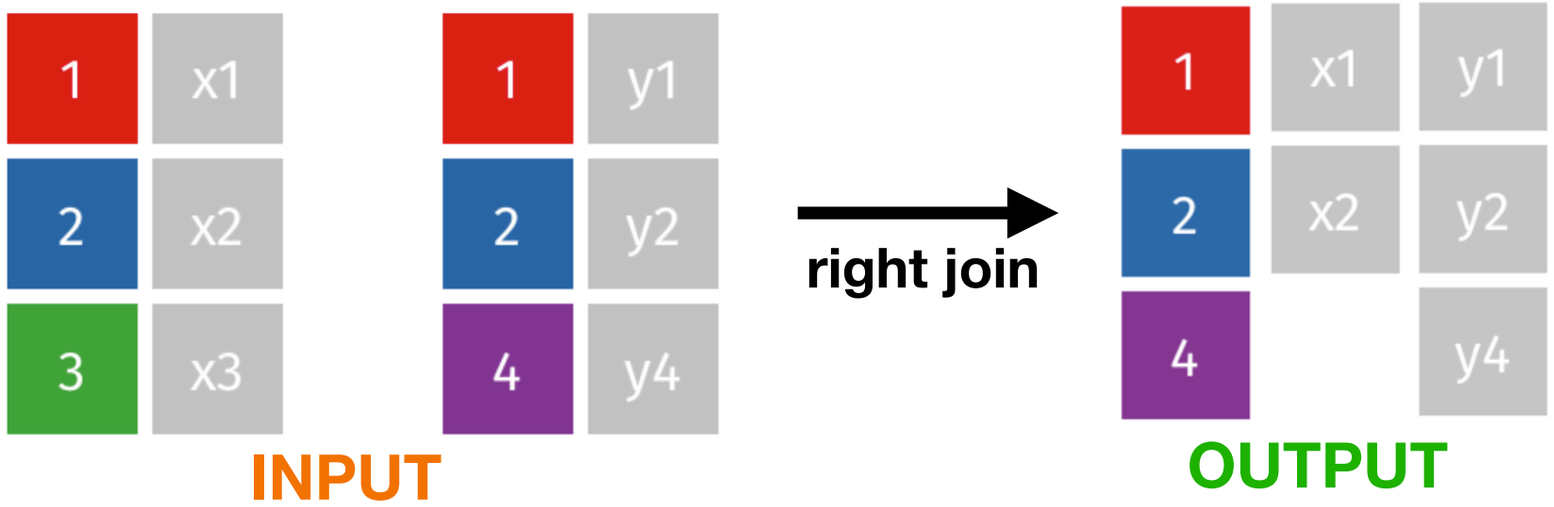
    x   y   w   z
0  x1  y1  w1  z1
1  x2  y2  w2  z2
2  x4  y4  w4  z4
```

```
LEFT JOIN:
    x   y    w    z
0  x1  y1   w1   z1
1  x2  y2   w2   z2
2  x3  y3  NaN  NaN
```

Right Joins



Select all records from Table B,
along with records from Table A
for which the join condition is met
(if at all).



EXAMPLE:

```
print("----JOINING DATAFRAMES---")
df1 = pd.DataFrame([['x1', 'y1'], ['x2', 'y2'], ['x3', 'y3']], columns=['x', 'y'])
df2 = pd.DataFrame([['x1', 'y1','w1', 'z1']
                  , [‘x2', 'y2', 'w2', 'z2']
                  , [‘x4', 'y4', 'w4', 'z4']],columns=['x', 'y','w', 'z'])
print("INPUTS:")
print(df1,'\n')
print(df2)

print("------")
print(‘RIGHT JOIN:  \n',df1.merge(df2, how=‘right'))
```

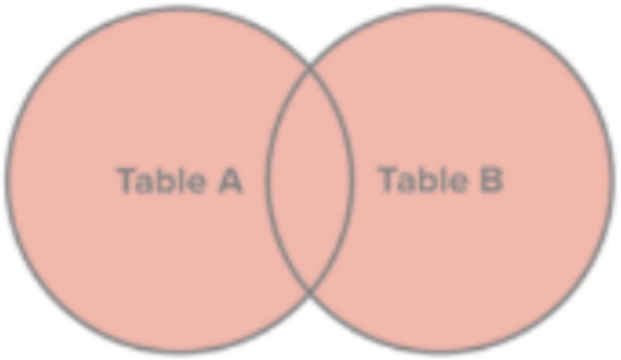
```
INPUTS:
    x   y
0  x1  y1
1  x2  y2
2  x3  y3

    x   y   w   z
0  x1  y1  w1  z1
1  x2  y2  w2  z2
2  x4  y4  w4  z4
```

```
RIGHT JOINS:
    x   y   w   z
0  x1  y1  w1  z1
1  x2  y2  w2  z2
2  x4  y4  w4  z4
```

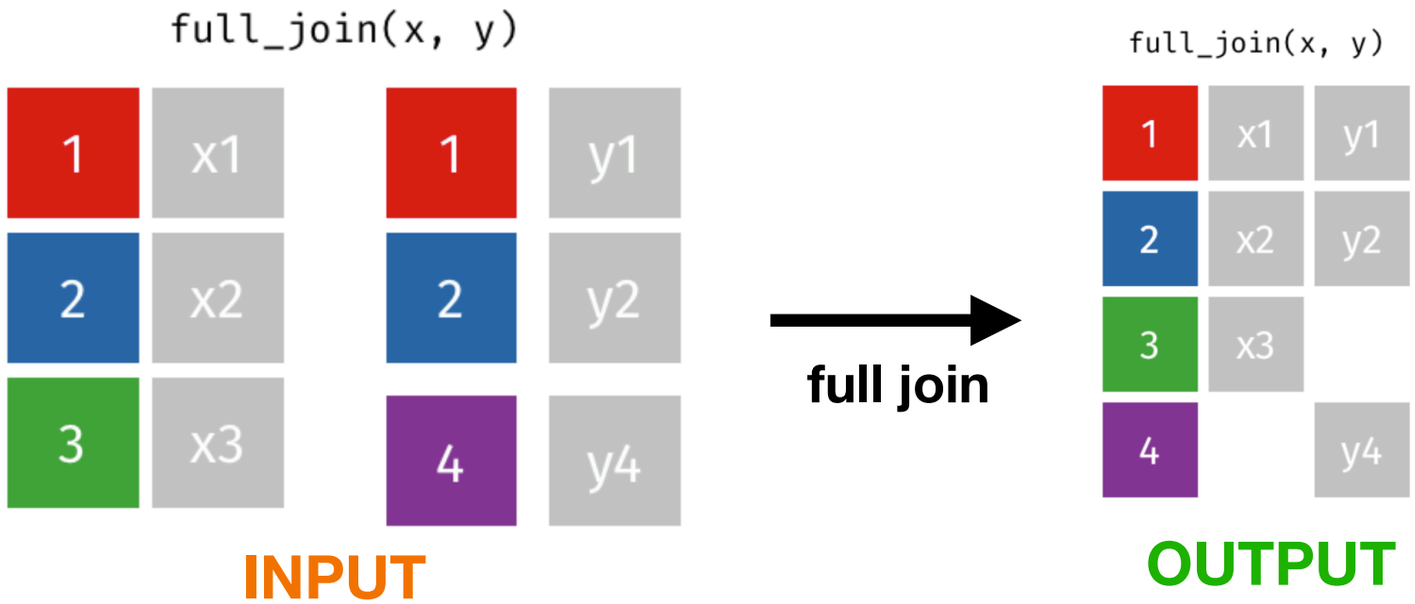
Full (outer) Joins

The union of two sets *A* and *B* is the set of elements which are in *A*, in *B*, or in both *A* and *B*.



Select all records from Table A and Table B, regardless of whether the join condition is met or not.

$$A \cup B = \{x : x \in A \textbf{ or } x \in B\}$$



EXAMPLE:

```
print("----JOINING DATAFRAMES---")
df1 = pd.DataFrame([['x1', 'y1'], ['x2', 'y2'], ['x3', 'y3']], columns=['x', 'y'])
df2 = pd.DataFrame([['x1', 'y1','w1', 'z1']
                    , [‘x2', 'y2', 'w2', 'z2']
                    , [‘x4', 'y4', 'w4', 'z4']],columns=['x', 'y','w', 'z'])
print("INPUTS:")
print(df1,'\n')
print(df2)

print("------")
print(‘OUTER JOIN:  \n',df1.merge(df2, how='outer'))
```

```
INPUTS:
    x   y
0  x1  y1
1  x2  y2
2  x3  y3

    x   y   w   z
0  x1  y1  w1  z1
1  x2  y2  w2  z2
2  x4  y4  w4  z4
```

```
OUTER JOIN:
     x    y    w    z
0   x1   y1   w1   z1
1   x2   y2   w2   z2
2   x3   y3  NaN  NaN
3   x4   y4   w4   z4
```

R Join examples

```r
library('tidyverse')

#LOAD DATA
df1 <- read_csv('data/toy-1.csv')
df2 <- read_csv('data/toy-2.csv')

print("----ORIGINAL----")
head(df1)
head(df2)

print("----INNER JOIN----")
df1 %>% inner_join(df2) %>% head()
df1 %>% inner_join(df2, by = c('x')) %>% head()
df1 %>% inner_join(df2, by = c('x' , 'w')) %>% head()

print("----LEFT JOIN----")
df1 %>% left_join(df2) %>% head()
# df1 %>% left_join(df2,by = c('x' , 'y')) %>% head()

print("----RIGHT JOIN----")
df1 %>% right_join(df2) %>% head()

print("----FULL JOIN----")
df1 %>% full_join(df2) %>% head()
```

OUTPUT

```
[1] "----ORIGINAL----"
# A tibble: 3 × 3
  x     y     w
  <chr> <chr> <chr>
1 x1    y1    w1
2 x2    y2    w2
3 x5    y5    w5
# A tibble: 4 × 4
  x     y     w     z
  <chr> <chr> <chr> <chr>
1 x1    y1    w1    z1
2 x2    y2    w2    z2
3 x3    y3    w3    z3
4 x4    y4    w4    z4
[1] "----INNER JOIN----"
Joining, by = c("x", "y", "w")
# A tibble: 2 × 4
  x     y     w     z
  <chr> <chr> <chr> <chr>
1 x1    y1    w1    z1
2 x2    y2    w2    z2
# A tibble: 2 × 6
  x     y.x   w.x   y.y   w.y   z
  <chr> <chr> <chr> <chr> <chr> <chr>
1 x1    y1    w1    y1    w1    z1
2 x2    y2    w2    y2    w2    z2
# A tibble: 2 × 5
  x     y.x   w     y.y   z
  <chr> <chr> <chr> <chr> <chr>
1 x1    y1    w1    y1    z1
2 x2    y2    w2    y2    z2
[1] "----LEFT JOIN----"
Joining, by = c("x", "y", "w")
# A tibble: 3 × 4
  x     y     w     z
  <chr> <chr> <chr> <chr>
1 x1    y1    w1    z1
2 x2    y2    w2    z2
3 x5    y5    w5    NA
[1] "----RIGHT JOIN----"
Joining, by = c("x", "y", "w")
# A tibble: 4 × 4
  x     y     w     z
  <chr> <chr> <chr> <chr>
1 x1    y1    w1    z1
2 x2    y2    w2    z2
3 x3    y3    w3    z3
4 x4    y4    w4    z4
[1] "----FULL JOIN----"
Joining, by = c("x", "y", "w")
# A tibble: 5 × 4
  x     y     w     z
  <chr> <chr> <chr> <chr>
1 x1    y1    w1    z1
2 x2    y2    w2    z2
3 x5    y5    w5    NA
4 x3    y3    w3    z3
5 x4    y4    w4    z4
```