



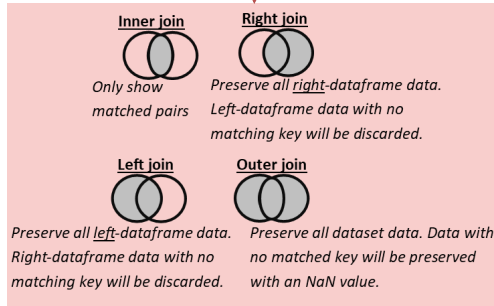
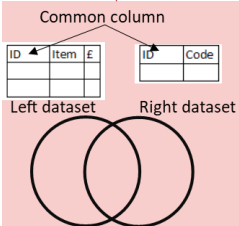
### Joining Dataframe:

```
pd.merge(left_df,right_df, on="col_name", how="***", suffixes("_left", "_right"))
```

Important to note which dataframe is left and right.

**"on="** tells pandas which col is the key col. key col being the unique col between the two datasets. use the key col to join the dataframes.

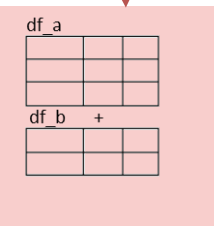
**"how="** can take values: "outer", "inner", "right" and "left" telling pandas what type of join it is. Remember the preserved datasets will keep all data. If you are preserving a dataset with no values, it will be NaN. More about joins below:



### Merging Datasets

#### Append Dataframe:

```
df=pd.concat([df_a,df_b])
```



### import

```
import pandas as pd
```

### New df

### View the df

#### #View the full df

```
df
```

#### #View the first or last data points

```
df.head() or df.tail()
```

#### #store number of rows and cols

```
row,col = df.shape
```

#### #See all the columns

```
list(df)
```

#### #Show certain rows:

```
df.iloc[10:20,0:2]#row,col
```

#### #Blank Dataframe:

```
df= pd.DataFrame(columns=["col1","col2"])  
#This is an empty Dataframe with 2 columns
```

#### #From a 2D list - with predefined column names:

```
data = [[1,2],  
        [3,4]]  
df = pd.DataFrame(data,columns=["col1","col2"])
```

#### #From a dictionary - dictionary keys are columns:

```
list1 = [1,2] #list of col1 data  
list2 = [3,4] #list of col2 data  
data = {"col1":list1, "col2":list2}  
df = pd.DataFrame(data)
```

#### #Read CSV into Dataframe (df)

```
df = pd.read_csv("data.csv", usecols=np.arange(cols))  
#cols = number of columns in your data
```

#### #Drop rows:

```
df = df.drop(row_list)
```

#### #Drop rows based on a condition

```
dfnew = df[df["col"] != "test"]  
# Selects rows that are *condition"
```

#### # Drop column

```
del df["col1"]  
# Or this:  
df = df.drop(["col1"], axis=1)
```

#### # Add new empty column

```
df["col_name"] = ""
```

#### #Re-arrange the columns

```
df = df["col1","col2"]
```

#### #Export full column into a list:

```
col_list = df["col1"].values.tolist()
```

#### #Set an already defined col as new index:

```
df= df.set_index('col')
```

#### #Create a MultiIndex using columns 'year' and 'month':

```
df = df.set_index(['year', 'month'])
```

#### #Reset Index

#Sometimes if we delete rows, the index will have jumps.  
#When we reset the index, the old index is added as a column  
#(we can drop this), and a new sequential index is used:

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

### Index Manipulation

### Pandas

### Clean the df

### Add data to df

### Accessing the Data

#### loc/iloc Preferred way to select data:

#### iloc

```
df.iloc[row,col]
```

Select by position:  
meaning you give it an integer value.

```
df.iloc[0,:]  
# : means all
```

```
df.iloc[[0,2,3,5,6]:]  
#locate data in given list and all col
```

```
df.iloc[[0:3],5]  
#locate data from given slice and col 5
```

#### loc

```
df.loc[row,col]
```

Select by label:  
meaning you give it actual column names "col1"

```
df.loc[0,"col1"]  
#row 0 and column "col1"
```

```
df.loc[2,["col1","col2"]]  
#row 2 and columns "col1" and "col2"
```

#### #Insert a row

```
df = df.append({}, ignore_index="false")
```

#Ignore index: TRUE will follow the index pattern. inside the {  
#you need a dictionary to define the data in your new row:

```
{"col1":list1, "col2":list2}  
#list1 = [list of col1 data] etc..
```

#### #Insert column with list of data:

```
df["NewCol"] = [1,2,3]
```

#### #Insert data to a particular cell:

```
df.loc[rownum,colnum] = "data"
```

#### Filtering Data - like excel column filters

#### # Sorting values (by a column. Exclude na position if not needed):

```
df.sort_values(by="col1", ascending=False, na_position="first")
```

#### # Filter rows based on a condition:

```
dfnew = df[df.col1 != "x"]
```

#### # Filter rows based on multiple conditions:

```
dfnew = df[ ( df.col1 != "x" ) & ( df.col2 >= 100 ) ]
```

