JOINING AND MERGING

In [1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Lets create 2 dictionaries for building a dataframe

In [2]:
```
names = pd.DataFrame({
'id' : [ 1 , 2 , 3 , 4 , 5 ],
'name' : [ 'Anna' , 'Bob' , 'Charles' ,
'Daniel' , 'Evan' ],
})
ages = pd.DataFrame({
'id' : [ 1 , 2 , 3 , 4 , 5 ],
'age' : [ 20 , 30 , 40 , 50 , 60 ]
})
```

Now when we have two separate data frames which are related to one another, we can combine them into one data frame. It is important that we have a common column that we can merge on. In this case, this is id .

In [3]:
```
df = pd.merge(names,ages, on = 'id' )
df.set_index( 'id' , inplace = True )
```

In [4]:
```
df
```

Out[4]:

id	name	age
1	Anna	20
2	Bob	30
3	Charles	40
4	Daniel	50
5	Evan	60

JOINS

It is not necessarily always obvious how we want to merge our data frames. This is where joins come into play. We have four types of joins.

JOIN MERGE TYPES	
JOIN	**DESCRIPTION**
left	Uses all keys from left object and merges with right
right	Uses all keys from right object and merges with left
outer	Uses all keys from both objects and merges them
inner	Uses only the keys which both objects have and merges them (default)

Now let's change our two data frames a little bit.

In [5]:
```
names = pd.DataFrame({
'id' : [ 1 , 2 , 3 , 4 , 5 , 6 ],
'name' : [ 'Anna' , 'Bob' , 'Charles' ,
'Daniel' , 'Evan' , 'Fiona' ],
})
ages = pd.DataFrame({
'id' : [ 1 , 2 , 3 , 4 , 5 , 7 ],
'age' : [ 20 , 30 , 40 , 50 , 60 , 70 ]
})
```

Our names frame now has an additional index 6 and an additional name. And our ages frame has an additional index 7 with an additional name.

In [6]:
```
df = pd.merge(names,ages, on = 'id' , how = 'inner' )
df.set_index( 'id' , inplace = True )
```

In [7]:
```
df
```

Out[7]:

id	name	age
1	Anna	20
2	Bob	30
3	Charles	40
4	Daniel	50
5	Evan	60

If we now perform the default inner join , we will end up with the same data frame as in the beginning. We only take the keys which both objects have. This means one to five.

In [8]:
```
df = pd.merge(names,ages, on = 'id' , how = 'left' )
df.set_index( 'id' , inplace = True )
```

In [9]:
```
df
```

Out[9]:

id	name	age
1	Anna	20.0
2	Bob	30.0
3	Charles	40.0
4	Daniel	50.0
5	Evan	60.0
6	Fiona	NaN

When we use the left join , we get all the keys from the names data frame but not the additional index 7 from ages. This also means that Fiona won't be assigned any age. The same principle goes for the right join just the other way around.

In [10]:
```
df = pd.merge(names,ages, on = 'id' , how = 'right' )
df.set_index( 'id' , inplace = True )
df
```

Out[10]:

id	name	age
1	Anna	20
2	Bob	30
3	Charles	40
4	Daniel	50
5	Evan	60
7	NaN	70

Now, we only have the keys from the ages frame and the 6 is missing. Finally, if we use the outer join , we combine all keys into one data frame.

In [11]:
```
df = pd.merge(names,ages, on = 'id' , how = 'outer' )
df.set_index( 'id' , inplace = True )
df
```

Out[11]:

id	name	age
1	Anna	20.0
2	Bob	30.0
3	Charles	40.0
4	Daniel	50.0
5	Evan	60.0
6	Fiona	NaN
7	NaN	70.0