In [1]:
```
import pandas as pd
```

Merge

In [2]:
```
df1 = pd.read_csv(r"C:\Users\kallzz\Desktop\Data Analytics Stuff\Data Analyst - Boot Camp\Python - Jupyter Notebooks\Datasets
df1
```

Out[2]:

	FellowshipID	FirstName	Skills
0	1001	Frodo	Hiding
1	1002	Samwise	Gardening
2	1003	Gandalf	Spells
3	1004	Pippin	Fireworks

In [3]:
```
d.read_csv(r"C:\Users\kallzz\Desktop\Data Analytics Stuff\Data Analyst - Boot Camp\Python - Jupyter Notebooks\Datasets\LOTR 2
```

Out[3]:

	FellowshipID	FirstName	Age
0	1001	Frodo	50
1	1002	Samwise	39
2	1006	Legolas	2931
3	1007	Elrond	6520
4	1008	Barromir	51

In [4]:
```
# merge function has defaults set to --> how = "inner"
df1.merge(df2)
```

Out[4]:

	FellowshipID	FirstName	Skills	Age
0	1001	Frodo	Hiding	50
1	1002	Samwise	Gardening	39

In [5]:
```
# how determines what kind of matching criteria for merging data between data frames
# how = 'inner' --> only matching data
# how = 'outer' --> all data matching and non-matching
# how = 'left'  --> all left df data with matching data in right df
# how = 'right' --> all right df data with matching data in left df

df1.merge(df2, how = 'inner')
```

Out[5]:

	FellowshipID	FirstName	Skills	Age
0	1001	Frodo	Hiding	50
1	1002	Samwise	Gardening	39

In [6]:
```
# we can specify on which column the data can be merged using 'on'
df1.merge(df2, how = 'inner', on = ['FellowshipID', 'FirstName'])
```

Out[6]:

	FellowshipID	FirstName	Skills	Age
0	1001	Frodo	Hiding	50
1	1002	Samwise	Gardening	39

In [7]:
```
# how = 'outer' --> all data matching and non-matching
df1.merge(df2, how = 'outer')
```

Out[7]:

	FellowshipID	FirstName	Skills	Age
0	1001	Frodo	Hiding	50.0
1	1002	Samwise	Gardening	39.0
2	1003	Gandalf	Spells	NaN
3	1004	Pippin	Fireworks	NaN
4	1006	Legolas	NaN	2931.0
5	1007	Elrond	NaN	6520.0
6	1008	Barromir	NaN	51.0

In [8]:
```
df1.merge(df2, how = 'left')
```

Out[8]:

	FellowshipID	FirstName	Skills	Age
0	1001	Frodo	Hiding	50.0
1	1002	Samwise	Gardening	39.0
2	1003	Gandalf	Spells	NaN
3	1004	Pippin	Fireworks	NaN

In [9]:
```
df1.merge(df2, how = 'right')
```

Out[9]:

	FellowshipID	FirstName	Skills	Age
0	1001	Frodo	Hiding	50
1	1002	Samwise	Gardening	39
2	1006	Legolas	NaN	2931
3	1007	Elrond	NaN	6520
4	1008	Barromir	NaN	51

In [10]:
```
# cross merge matches 1-1 mapping between data frames
df1.merge(df2, how = 'cross')
```

Out[10]:

	FellowshipID_x	FirstName_x	Skills	FellowshipID_y	FirstName_y	Age
0	1001	Frodo	Hiding	1001	Frodo	50
1	1001	Frodo	Hiding	1002	Samwise	39
2	1001	Frodo	Hiding	1006	Legolas	2931
3	1001	Frodo	Hiding	1007	Elrond	6520
4	1001	Frodo	Hiding	1008	Barromir	51
5	1002	Samwise	Gardening	1001	Frodo	50
6	1002	Samwise	Gardening	1002	Samwise	39
7	1002	Samwise	Gardening	1006	Legolas	2931
8	1002	Samwise	Gardening	1007	Elrond	6520
9	1002	Samwise	Gardening	1008	Barromir	51
10	1003	Gandalf	Spells	1001	Frodo	50
11	1003	Gandalf	Spells	1002	Samwise	39
12	1003	Gandalf	Spells	1006	Legolas	2931
13	1003	Gandalf	Spells	1007	Elrond	6520
14	1003	Gandalf	Spells	1008	Barromir	51
15	1004	Pippin	Fireworks	1001	Frodo	50
16	1004	Pippin	Fireworks	1002	Samwise	39
17	1004	Pippin	Fireworks	1006	Legolas	2931
18	1004	Pippin	Fireworks	1007	Elrond	6520
19	1004	Pippin	Fireworks	1008	Barromir	51

******* **JOIN** ********

In [14]:
```
df1.join(df2)
```

```
    739         left_indexer,
  (...)
    744         use_na_proxy=True,
    745     )

File ~\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:2458, in _items_overlap_with_suffix(left, right, suffixes)
   2455 lsuffix, rsuffix = suffixes
   2457 if not lsuffix and not rsuffix:
-> 2458     raise ValueError(f"columns overlap but no suffix specified: {to_rename}")
   2460 def renamer(x, suffix):
   2461     """
   2462     Rename the left and right indices.
   2463
  (...)
   2474     x : renamed column name
   2475     """

ValueError: columns overlap but no suffix specified: Index(['FellowshipID', 'FirstName'], dtype='object')
```

** Merge Vs Join **

In [15]:
```
# merge can work with defaults and easy to operate
# join needs explicit options for 'how' and 'on' attributes
# all the remaining functionality is same
# all types joins can be performed
df1.join(df2, how = 'outer', on = ['FellowshipID', 'FirstName'])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[15], line 5
      1 # merge can work with defaults and easy to operate
      2 # join needs explicit options for 'how' and 'on' attributes
      3 # all the remaining functionality is same
      4 # all types joins can be performed
----> 5 df1.join(df2, how = 'outer', on = ['FellowshipID', 'FirstName'])

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:9979, in DataFrame.join(self, other, on, how, lsuffix, rsuffix, sor
t, validate)
   9816 def join(
   9817     self,
   9818     other: DataFrame | Series | list[DataFrame | Series],
   (...)
   9824     validate: str | None = None,
   9825 ) -> DataFrame:
   9826     """
   9827     Join columns of another DataFrame.
   9828
   (...)
   9977     5  K1  A5   B1
   9978     """
-> 9979     return self._join_compat(
   9980         other,
   9981         on=on,
   9982         how=how,
   9983         lsuffix=lsuffix,
   9984         rsuffix=rsuffix,
   9985         sort=sort,
   9986         validate=validate,
   9987     )

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:10018, in DataFrame._join_compat(self, other, on, how, lsuffix, rsu
ffix, sort, validate)
  10008     if how == "cross":
  10009         return merge(
  10010             self,
  10011             other,
   (...)
  10016             validate=validate,
  10017         )
> 10018     return merge(
  10019         self,
  10020         other,
  10021         left_on=on,
  10022         how=how,
  10023         left_index=on is None,
  10024         right_index=True,
  10025         suffixes=(lsuffix, rsuffix),
  10026         sort=sort,
  10027         validate=validate,
  10028     )
  10029 else:
  10030     if on is not None:

File ~\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:110, in merge(left, right, how, on, left_on, right_on, left
_index, right_index, sort, suffixes, copy, indicator, validate)
     93 @Substitution("\nleft : DataFrame or named Series")
     94 @Appender(_merge_doc, indents=0)
     95 def merge(
   (...)
    108     validate: str | None = None,
    109 ) -> DataFrame:
--> 110     op = _MergeOperation(
    111         left,
    112         right,
    113         how=how,
    114         on=on,
    115         left_on=left_on,
    116         right_on=right_on,
    117         left_index=left_index,
    118         right_index=right_index,
    119         sort=sort,
    120         suffixes=suffixes,
    121         indicator=indicator,
    122         validate=validate,
    123     )
    124     return op.get_result(copy=copy)

File ~\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:685, in _MergeOperation.__init__(self, left, right, how, o
n, left_on, right_on, axis, left_index, right_index, sort, suffixes, indicator, validate)
    681     # stacklevel chosen to be correct when this is reached via pd.merge
    682     # (and not DataFrame.join)
    683     warnings.warn(msg, FutureWarning, stacklevel=find_stack_level())
--> 685 self.left_on, self.right_on = self._validate_left_right_on(left_on, right_on)
    687 cross_col = None
    688 if self.how == "cross":

File ~\anaconda3\lib\site-packages\pandas\core\reshape\merge.py:1469, in _MergeOperation._validate_left_right_on(self, left
_on, right_on)
   1467     if self.right_index:
```

```
   1468         if len(left_on) != self.right.index.nlevels:
-> 1469             raise ValueError(
   1470                 "len(left_on) must equal the number "
   1471                 'of levels in the index of "right"'
   1472             )
   1473         right_on = [None] * n
   1474 elif right_on is not None:

ValueError: len(left_on) must equal the number of levels in the index of "right"
```

In [18]:
```
df1.join(df2, how = 'outer', on = 'FellowshipID', lsuffix = '_Left', rsuffix = '_Right')
```

Out[18]:

	FellowshipID	FellowshipID_Left	FirstName_Left	Skills	FellowshipID_Right	FirstName_Right	Age
0.0	1001	1001.0	Frodo	Hiding	NaN	NaN	NaN
1.0	1002	1002.0	Samwise	Gardening	NaN	NaN	NaN
2.0	1003	1003.0	Gandalf	Spells	NaN	NaN	NaN
3.0	1004	1004.0	Pippin	Fireworks	NaN	NaN	NaN
NaN	0	NaN	NaN	NaN	1001.0	Frodo	50.0
NaN	1	NaN	NaN	NaN	1002.0	Samwise	39.0
NaN	2	NaN	NaN	NaN	1006.0	Legolas	2931.0
NaN	3	NaN	NaN	NaN	1007.0	Elrond	6520.0
NaN	4	NaN	NaN	NaN	1008.0	Barromir	51.0

Concatenate

- Only can inner (intersect) or outer (union) join the other axis
- It concatenates the data one on the other. Merge merges the data side by side

In [19]:
```
pd.concat([df1,df2])
```

Out[19]:

	FellowshipID	FirstName	Skills	Age
0	1001	Frodo	Hiding	NaN
1	1002	Samwise	Gardening	NaN
2	1003	Gandalf	Spells	NaN
3	1004	Pippin	Fireworks	NaN
0	1001	Frodo	NaN	50.0
1	1002	Samwise	NaN	39.0
2	1006	Legolas	NaN	2931.0
3	1007	Elrond	NaN	6520.0
4	1008	Barromir	NaN	51.0

In [20]:
```
pd.concat([df1,df2], join = 'inner')
```

Out[20]:

	FellowshipID	FirstName
0	1001	Frodo
1	1002	Samwise
2	1003	Gandalf
3	1004	Pippin
0	1001	Frodo
1	1002	Samwise
2	1006	Legolas
3	1007	Elrond
4	1008	Barromir

In [25]:
```
# pd.concat([df1,df2], join = 'outer')
# axis = 0 is the default value
pd.concat([df1,df2], join = 'outer', axis = 0)
```

Out[25]:

	FellowshipID	FirstName	Skills	Age
0	1001	Frodo	Hiding	NaN
1	1002	Samwise	Gardening	NaN
2	1003	Gandalf	Spells	NaN
3	1004	Pippin	Fireworks	NaN
0	1001	Frodo	NaN	50.0
1	1002	Samwise	NaN	39.0
2	1006	Legolas	NaN	2931.0
3	1007	Elrond	NaN	6520.0
4	1008	Barromir	NaN	51.0

In [26]:
```
pd.concat([df1,df2], join = 'outer', axis = 1)
```

Out[26]:

	FellowshipID	FirstName	Skills	FellowshipID	FirstName	Age
0	1001.0	Frodo	Hiding	1001	Frodo	50
1	1002.0	Samwise	Gardening	1002	Samwise	39
2	1003.0	Gandalf	Spells	1006	Legolas	2931
3	1004.0	Pippin	Fireworks	1007	Elrond	6520
4	NaN	NaN	NaN	1008	Barromir	51

In [23]:
```
pd.concat([df1,df2], join = 'left')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[23], line 1
----> 1 pd.concat([df1,df2], join = 'left')

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:331, in deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\core\reshape\concat.py:368, in concat(objs, axis, join, ignore_index, keys, levels, names, verify_integrity, sort, copy)
    146 @deprecate_nonkeyword_arguments(version=None, allowed_args=["objs"])
    147 def concat(
    148     objs: Iterable[NDFrame] | Mapping[HashableT, NDFrame],
   (...)
    157     copy: bool = True,
    158 ) -> DataFrame | Series:
    159     """
    160     Concatenate pandas objects along a particular axis.
    161
   (...)
    366     1   3   4
    367     """
--> 368     op = _Concatenator(
    369         objs,
    370         axis=axis,
    371         ignore_index=ignore_index,
    372         join=join,
    373         keys=keys,
    374         levels=levels,
    375         names=names,
    376         verify_integrity=verify_integrity,
    377         copy=copy,
    378         sort=sort,
    379     )
    381     return op.get_result()

File ~\anaconda3\lib\site-packages\pandas\core\reshape\concat.py:413, in _Concatenator.__init__(self, objs, axis, join, keys, levels, names, ignore_index, verify_integrity, copy, sort)
    411     self.intersect = True
    412 else:  # pragma: no cover
--> 413     raise ValueError(
    414         "Only can inner (intersect) or outer (union) join the other axis"
    415     )
    417 if isinstance(objs, abc.Mapping):
    418     if keys is None:

ValueError: Only can inner (intersect) or outer (union) join the other axis
```

In [ ]:
```

```