

pd.concat()

```
pd.concat([df1, df2])
```

```
pd.concat([df1, df2], keys=['1', '2'])
```

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

```
df1.append(df2)
```

Multi-indexing

```
pd.MultiIndex.from_arrays([['a', 'a', 'b', 'b'], [1, 2, 1, 2]])
```

```
pd.MultiIndex.from_tuples([('a', 1), ('a', 2), ('b', 1), ('b', 2)])
```

```
pd.MultiIndex.from_product([['a', 'b'], [1, 2]])
```

```
df.sort_index()
```

Orders index in ascending order

```
df.reset_index(name='')
```

Turn the index labels into cols

All three methods of `pd.MultiIndex` give:

```
MultiIndex([(a, 1), (a, 2), (b, 1), (b, 2)])
```

Joins

```
pd.merge(df1, df2)
```

```
pd.merge(df1, df2, left_on="a", right_on="b")
```

Creates a label (key) for each df

Default join is "inner join"

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

Does the same thing as

```
pd.concat()
```

```
pd.merge(df1, df2, on="a", suffixes=["_L", "_R"])
```

Aggregation and Grouping (cont)

```
df.groupby('key').transform()
```

```
df.groupby('key').agg()
```

Merges data using "a" and "b" as keys

```
df.groupby('key').apply(func)
```

Default is "outer join"

When multiple

Pivot Tables

```
df.pivot_table(index='a', columns='b', values='sum')
```

cols have same index and name, gives suffix "_L" and "_R"

```
df.pivot_table(index='a', columns='b', values='sum').plot()
```

Calculates the mean across cols

Gives sum for each key

Pivot tables do the same thing as using `groupby` with aggregation functions. The main difference is that they are a cleaner way of using multiple aggregation functions at once for a single grouping

Gives summary stats

Aggregation and Grouping

```
df.mean(axis='columns')
```

```
df.groupby('key').sum()
```

```
df.groupby('key')['a'].sum()
```

```
df.groupby('key')['a'].describe()
```



By robeyjorgensen

Published 10th October, 2020.

Last updated 10th October, 2020.

Page 1 of 1.

Sponsored by [CrosswordCheats.com](https://crosswordcheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>