

## Database-style Operations on Dataframes

**Background on the data** Data meanings:

- PRCP : precipitation in millimeters
- SNOW : snowfall in millimeters
- SNWD : snow depth in millimeters
- TMAX : maximum daily temperature in Celsius
- TMIN : minimum daily temperature in Celsius
- TOBS : temperature at time of observation in Celsius
- WESF : water equivalent of snow in millimeters

### Setup

```
import pandas as pd
weather = pd.read_csv('data/nyc_weather_2018.csv')
weather.head()
```

	attributes	datatype	date	station	value	
0	„N,	PRCP	2018-01-01T00:00:00	GHCND:US1CTFR0039	0.0	
1	„N,	PRCP	2018-01-01T00:00:00	GHCND:US1NJBG0015	0.0	
2	„N,	SNOW	2018-01-01T00:00:00	GHCND:US1NJBG0015	0.0	
3	„N,	PRCP	2018-01-01T00:00:00	GHCND:US1NJBG0017	0.0	
4	„N,	SNOW	2018-01-01T00:00:00	GHCND:US1NJBG0017	0.0	

Next steps: [View recommended plots](#)

## Querying DataFrames

The `query()` method is an easier way of filtering based on some criteria. For example, we can use it to find all entries where snow was recorded:

```
snow_data = weather.query('datatype == "SNOW" and value > 0') #filtering based on some criteria using query()
snow_data.head()
```

	attributes	datatype	date	station	value	
124	„N,	SNOW	2018-01-01T00:00:00	GHCND:US1NYWC0019	25.0	
723	„N,	SNOW	2018-01-04T00:00:00	GHCND:US1NJBG0015	229.0	
726	„N,	SNOW	2018-01-04T00:00:00	GHCND:US1NJBG0017	10.0	
730	„N,	SNOW	2018-01-04T00:00:00	GHCND:US1NJBG0018	46.0	
737	„N,	SNOW	2018-01-04T00:00:00	GHCND:US1NJES0018	10.0	

```
import sqlite3
with sqlite3.connect('data/weather.db') as connection:

    # Establish connection to the SQLite database
    snow_data_from_db = pd.read_sql(
        'SELECT * FROM weather WHERE datatype == "SNOW" AND value > 0',
        connection
    )
    snow_data.reset_index().drop(columns='index').equals(snow_data_from_db)

    True
```

Note this is also equivalent to creating Boolean masks:

```
weather[(weather.datatype == 'SNOW') & (weather.value > 0)].equals(snow_data)

True
```

## ✓ Merging DataFrames

We have data for many different stations each day; however, we don't know what the stations are just their IDs. We can join the data in the `data/weather_stations.csv` file which contains information from the stations endpoint of the NCEI API. Consult the `weather_data_collection.ipynb` notebook to see how this was collected. It looks like this:

```
station_info = pd.read_csv('data/weather_stations.csv')
station_info.head()
```

	id	name	latitude	longitude	elevation	
0	GHCND:US1CTFR0022	STAMFORD 2.6 SSW, CT US	41.0641	-73.5770	36.6	
1	GHCND:US1CTFR0039	STAMFORD 4.2 S, CT US	41.0378	-73.5682	6.4	
2	GHCND:US1NJBG0001	BERGENFIELD 0.3 SW, NJ US	40.9213	-74.0020	20.1	
3	GHCND:US1NJBG0002	SADDLE BROOK TWP 0.6 E, NJ US	40.9027	-74.0834	16.8	

Next steps: [View recommended plots](#)

```
weather.head()
```

	attributes	datatype	date	station	value	
0	„N,	PRCP	2018-01-01T00:00:00	GHCND:US1CTFR0039	0.0	
1	„N,	PRCP	2018-01-01T00:00:00	GHCND:US1NJBG0015	0.0	
2	„N,	SNOW	2018-01-01T00:00:00	GHCND:US1NJBG0015	0.0	
3	„N,	PRCP	2018-01-01T00:00:00	GHCND:US1NJBG0017	0.0	
4	„N,	SNOW	2018-01-01T00:00:00	GHCND:US1NJBG0017	0.0	

Next steps: [View recommended plots](#)

We can join our data by matching up the `station_info.id` column with the `weather.station` column. Before doing that though, let's see how many unique values we have:

```
station_info.id.describe() #info
```

```
count          262
unique          262
top    GHCND:US1CTFR0022
freq           1
Name: id, dtype: object
```

While `station_info` has one row per station, the `weather` dataframe has many entries per station. Notice it also has fewer uniques:

```
weather.station.describe() #Info
```

```
count          80256
unique          109
top    GHCND:USW00094789
freq          4270
Name: station, dtype: object
```

When working with joins, it is important to keep an eye on the row count. Some join types will lead to data loss:

```
station_info.shape[0], weather.shape[0]
```

```
(262, 80256)
```

Since we will be doing this often, it makes more sense to write a function:

```
def get_row_count(*dfs):
    # Return a list containing the number of rows for each DataFrame
    return [df.shape[0] for df in dfs]

# This will return a list containing the number of rows for each DataFrame
get_row_count(station_info, weather)

[262, 80256]
```

The `map()` function is more efficient than list comprehensions. We can couple this with `getattr()` to grab any attribute for multiple dataframes:

```
def get_info(attr, *dfs):
    # Use a lambda function to apply the getattr function to each DataFrame in the list of DataFrames (dfs)
    return list(map(lambda x: getattr(x, attr), dfs))

# This will return a list containing the shapes (number of rows and columns) of each DataFrame
get_info('shape', station_info, weather)

[(262, 5), (80256, 5)]
```

By default `merge()` performs an inner join. We simply specify the columns to use for the join. The left dataframe is the one we call `merge()` on, and the right one is passed in as an argument:

```
# Perform an inner join operation between the weather and station_info DataFrames
inner_join = weather.merge(station_info, left_on='station', right_on='id')

# Sample 5 random rows from the resulting DataFrame (inner_join)
# The 'random_state=0' parameter ensures reproducibility of the random sampling
inner_join.sample(5, random_state=0)
```

	attributes	datatype	date	station	value	
27422	„N,	PRCP	2018-01-23T00:00:00	GHCND:US1NYSF0061	2.3	GHCND:US1NYSF0061
19317	T,„N,	PRCP	2018-08-10T00:00:00	GHCND:US1NJUN0014	0.0	GHCND:US1NJUN0014
13778	„N,	WESF	2018-02-18T00:00:00	GHCND:US1NJMS0089	19.6	GHCND:US1NJMS0089

We can remove the duplication of information in the station and id columns by renaming one of them before the merge and then simply using `on`

```
weather.merge(station_info.rename(dict(id='station')), axis=1, on='station').sample(5, random_state=0)
```

	attributes	datatype	date	station	value	name	1
27422	„N,	PRCP	2018-01-23T00:00:00	GHCND:US1NYSF0061	2.3	CENTERPORT 0.9 SW, NY US	
19317	T,„N,	PRCP	2018-08-10T00:00:00	GHCND:US1NJUN0014	0.0	WESTFIELD 0.6 NE, NJ US	
13778	„N,	WESF	2018-02-18T00:00:00	GHCND:US1NJMS0089	19.6	PARSIPPANY TROY HILLS TWP 1 3 N 11 US	

We are losing stations that don't have weather observations associated with them, if we don't want to lose these rows, we perform a right or left join instead of the inner join

```
left_join = station_info.merge(weather, left_on='id', right_on='station', how='left')
right_join = weather.merge(station_info, left_on='station', right_on='id', how='right')
right_join.tail()
```

	attributes	datatype	date	station	value	
80404	„W,	WDF5	2018-12-31T00:00:00	GHCND:USW00094789	130.0	GHCND:USW000947
80405	„W,	WSF2	2018-12-31T00:00:00	GHCND:USW00094789	9.8	GHCND:USW000947

The left and right join as we performed above are equivalent because the side that we kept the rows without matches was the same in both cases:

```
left_join.sort_index(axis=1).sort_values(['date', 'station']).reset_index().drop(columns='index').equals(
right_join.sort_index(axis=1).sort_values(['date', 'station']).reset_index().drop(columns='index')
)
```

True

Note we have additional rows in the left and right joins because we kept all the stations that didn't have weather observations:

```
get_info('shape', inner_join, left_join, right_join)
```

```
[(80256, 10), (80409, 10), (80409, 10)]
```

```
outer_join = weather.merge(
station_info[station_info.name.str.contains('NY')],
left_on='station', right_on='id', how='outer', indicator=True
)
outer_join.sample(4, random_state=0).append(outer_join[outer_join.station.isna()].head(2))
```

```
<ipython-input-22-81b63e73e04e>:5: FutureWarning: The frame.append method is deprecated
outer_join.sample(4, random_state=0).append(outer_join[outer_join.station.isna()].head(2))
```

	attributes	datatype	date	station	value	
17259	„N,	PRCP	2018-05-15T00:00:00	GHCND:US1NJPS0022	0.3	N
76178	„N,	PRCP	2018-05-19T00:00:00	GHCND:US1NJPS0015	8.1	N
73410	„N,	MDPR	2018-08-05T00:00:00	GHCND:US1NYNS0018	12.2	GHCND:US1NYNS0018

```
import sqlite3
with sqlite3.connect('data/weather.db') as connection:
    inner_join_from_db = pd.read_sql(
        'SELECT * FROM weather JOIN stations ON weather.station == stations.id',
        connection
    )
inner_join_from_db.shape == inner_join.shape
```

True

```
dirty_data = pd.read_csv(
'data/dirty_data2.csv', index_col='date'
).drop_duplicates().drop(columns='SNWD')
dirty_data.head()
```

	station	PRCP	SNOW	TMAX	TMIN	TOBS	WESF	inclement_weathe
date								
2018-01-01T00:00:00	?	0.0	0.0	5505.0	-40.0	NaN	NaN	Na
2018-01-02T00:00:00	GHCND:USC00280907	0.0	0.0	-8.3	-16.1	-12.2	NaN	Fals
2018-01-03T00:00:00	GHCND:USC00280907	0.0	0.0	-4.4	-13.9	-13.3	NaN	Fals

Next steps:

[View recommended plots](#)

```
valid_station = dirty_data.query('station != "?"').copy().drop(columns=['WESF', 'station'])
station_with_wesf = dirty_data.query('station == "?"').copy().drop(columns=['station', 'TOBS', 'TMIN', 'TMAX'])
```

```
valid_station.merge(
    station_with_wesf, left_index=True, right_index=True
).query('WESF > 0').head()
```

	PRCP_x	SNOW_x	TMAX	TMIN	TOBS	inclement_weather_x	PRCP_y	SNOW_y	WESF
date									
2018-01-30T00:00:00	0.0	0.0	6.7	-1.7	-0.6	False	1.5	13.0	1.8
2018-03-08T00:00:00	48.8	NaN	1.1	-0.6	1.1	False	28.4	NaN	28.7
2018-03-13T00:00:00	4.1	51.0	5.6	-3.9	0.0	True	3.0	13.0	3.0

```
valid_station.merge(
    station_with_wesf, left_index=True, right_index=True, suffixes=('_', '_?')
).query('WESF > 0').head()
```

	PRCP	SNOW	TMAX	TMIN	TOBS	inclement_weather	PRCP_?	SNOW_?	WESF	incl
date										
2018-01-30T00:00:00	0.0	0.0	6.7	-1.7	-0.6	False	1.5	13.0	1.8	
2018-03-08T00:00:00	48.8	NaN	1.1	-0.6	1.1	False	28.4	NaN	28.7	
2018-03-13T00:00:00	4.1	51.0	5.6	-3.9	0.0	True	3.0	13.0	3.0	

```
valid_station.join(station_with_wesf, rsuffix='_?').query('WESF > 0').head()
```

	PRCP	SNOW	TMAX	TMIN	TOBS	inclement_weather	PRCP_?	SNOW_?	WESF	incl
date										
2018-01-30T00:00:00	0.0	0.0	6.7	-1.7	-0.6	False	1.5	13.0	1.8	
2018-03-08T00:00:00	48.8	NaN	1.1	-0.6	1.1	False	28.4	NaN	28.7	
2018-03-13T00:00:00	4.1	51.0	5.6	-3.9	0.0	True	3.0	13.0	3.0	

```
weather.set_index('station', inplace=True)
station_info.set_index('id', inplace=True)
```

```
weather.index.intersection(station_info.index)
```

```
Index(['GHCND:US1CTFR0039', 'GHCND:US1NJBG0015', 'GHCND:US1NJBG0017',
      'GHCND:US1NJBG0018', 'GHCND:US1NJBG0023', 'GHCND:US1NJBG0030',
```

```
'GHCND:US1NJBG0039', 'GHCND:US1NJBG0044', 'GHCND:US1NYES0018',
'GHCND:US1NYES0024',
...
'GHCND:US1NJMS0047', 'GHCND:US1NYSF0083', 'GHCND:US1NINY0074',
'GHCND:US1NJPS0018', 'GHCND:US1NJBG0037', 'GHCND:USC00284987',
'GHCND:US1NYES0031', 'GHCND:US1NJMD0086', 'GHCND:US1NJMS0097',
'GHCND:US1NJMN0081'],
dtype='object', length=109)
```

```
weather.index.difference(station_info.index)
```

```
Index([], dtype='object')
```

```
station_info.index.difference(weather.index)
```

```
Index(['GHCND:US1CTFR0022', 'GHCND:US1NJBG0001', 'GHCND:US1NJBG0002',
'GHCND:US1NJBG0005', 'GHCND:US1NJBG0006', 'GHCND:US1NJBG0008',
'GHCND:US1NJBG0011', 'GHCND:US1NJBG0012', 'GHCND:US1NJBG0013',
'GHCND:US1NJBG0020',
...
'GHCND:USC00308322', 'GHCND:USC00308749', 'GHCND:USC00308946',
'GHCND:USC00309117', 'GHCND:USC00309270', 'GHCND:USC00309400',
'GHCND:USC00309466', 'GHCND:USC00309576', 'GHCND:USW00014708',
'GHCND:USW00014786'],
dtype='object', length=153)
```

```
ny_in_name = station_info[station_info.name.str.contains('NY')]
```

```
ny_in_name.index.difference(weather.index).shape[0]\
+ weather.index.difference(ny_in_name.index).shape[0]\
== weather.index.symmetric_difference(ny_in_name.index).shape[0]
```

```
True
```

```
weather.index.unique().union(station_info.index)
```

```
Index(['GHCND:US1CTFR0022', 'GHCND:US1CTFR0039', 'GHCND:US1NJBG0001',
'GHCND:US1NJBG0002', 'GHCND:US1NJBG0003', 'GHCND:US1NJBG0005',
'GHCND:US1NJBG0006', 'GHCND:US1NJBG0008', 'GHCND:US1NJBG0010',
'GHCND:US1NJBG0011',
...
'GHCND:USW00014708', 'GHCND:USW00014732', 'GHCND:USW00014734',
'GHCND:USW00014786', 'GHCND:USW00054743', 'GHCND:USW00054787',
'GHCND:USW00094728', 'GHCND:USW00094741', 'GHCND:USW00094745',
'GHCND:USW00094789'],
dtype='object', length=262)
```

```
ny_in_name = station_info[station_info.name.str.contains('NY')]
ny_in_name.index.difference(weather.index).union(weather.index.difference(ny_in_name.index)).equals(
weather.index.symmetric_difference(ny_in_name.index)
)
```

```
True
```