```
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

```
# using the .query() method
snow_data = weather.query('datatype == "SNOW" and value > 0') # positive values of snow datatype
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: # uploaded weather.db to colab
  snow_data_from_db = pd.read_sql(
    'SELECT * FROM weather WHERE datatype == "SNOW" AND value > 0', # similar to query above
    connection
  )
snow_data.reset_index().drop(columns='index').equals(snow_data_from_db) # check if the output is similar to the query above
```

```
True
```

```
weather[(weather.datatype == 'SNOW') & (weather.value > 0)].equals(snow_data) # also similar to this code
```

```
True
```

```
station_info = pd.read_csv('data/weather_stations.csv') # uploaded to colab
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
4	GHCND:US1NJBG0003	TENAFLY 1.3 W, NJ US	40.9147	-73.9775	21.6

Next steps: View recommended plots

```
weather.head() # recall that weather data looks like this
# what we aim is for the station to be matched with the station name from weather_stations.csv
```

	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

```
station_info.id.describe() # looks at how many unique values are there
#could also use .unique()
```

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

```
weather.station.describe()  # there are only 109 unique stations for the weather data
```

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

```
station_info.shape[0], weather.shape[0]
# notice that there are 262 rows for station info compared to the 80,256 rows of weather
```

```
(262, 80256)
```

```
def get_row_count(*dfs): # creates a function to check the row count
  return [df.shape[0] for df in dfs] # dfs mean multiple dataframes
get_row_count(station_info, weather)
```

```
[262, 80256]
```

```
def get_info(attr, *dfs): # more general function
  return list(map(lambda x: getattr(x, attr), dfs))
  # performs the lambda function to every dataframe
get_info('shape', station_info, weather) # gets the row count of station info and weather
```

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

```
inner_join = weather.merge(station_info, left_on='station', right_on='id') # performs inner join
inner_join.sample(5, random_state=0) # gets 5 datapoints
```

	attributes	datatype	date	station	value	id
27422	„N,	PRCP	2018-01-23T00:00:00	GHCND:US1NYSF0061	2.3	GHCND:US1NYSF006
19317	T,„N,	PRCP	2018-08-10T00:00:00	GHCND:US1NJUN0014	0.0	GHCND:US1NJUN001
13778	„N,	WESF	2018-02-18T00:00:00	GHCND:US1NJMS0089	19.6	GHCND:US1NJMS008

```
weather.merge(station_info.rename(dict(id='station'), axis=1), on='station').sample(5, random_state=0)
# renames the id to station, then merges them and gets 5 datapoints
```

	attributes	datatype	date	station	value	name	lat
27422	„N,	PRCP	2018-01-23T00:00:00	GHCND:US1NYSF0061	2.3	CENTERPORT 0.9 SW, NY US	4(
19317	T„N,	PRCP	2018-08-10T00:00:00	GHCND:US1NJUN0014	0.0	WESTFIELD 0.6 NE, NJ US	4(
13778	„N,	WESF	2018-02-18T00:00:00	GHCND:US1NJMS0089	19.6	PARSIPPANY TROY HILLS TWP 1.3, NJ US	4(

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

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

```
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')
)
# checks if the right join is similar to the left join
```

```
True
```

```
get_info('shape', inner_join, left_join, right_join)
# note that there are 153 more data points with the right and left join compared to the inner join
```

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

```
outer_join = weather.merge( # performs an outer join
  station_info[station_info.name.str.contains('NY')], # gets station name with 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-22-1ad5c7ccb9c6>:5: FutureWarning: The frame.append method is deprecated
  outer_join.sample(4, random_state=0).append(outer_join[outer_join.station.isna()].head
```

	attributes	datatype	date	station	value	id
17259	„N,	PRCP	2018-05-15T00:00:00	GHCND:US1NJPS0022	0.3	NaN
76178	„N,	PRCP	2018-05-19T00:00:00	GHCND:US1NJPS0015	8.1	NaN
73410	„N,	MDPR	2018-08-05T00:00:00	GHCND:US1NYNS0018	12.2	GHCND:US1NYNS001

```
import sqlite3 # inner join equivalent using sqlite

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 # checks if the inner joins were equivalent
```

```
True
```

```
dirty_data = pd.read_csv( # reading the dirty data from module 7
  'data/dirty_data.csv', index_col='date'
).drop_duplicates().drop(columns='SNWD')
dirty_data.head()
```

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

Next steps: View recommended plots

```
# removes data with ?, and the WESF and station column
valid_station = dirty_data.query('station != "?"').copy().drop(columns=['WESF', 'station'])
# gets the data with ?, removes the columns in the .drop() method
station_with_wesf = dirty_data.query('station == "?"').copy().drop(columns=['station', 'TOBS', 'TMIN', 'TMAX'])


valid_station.merge( # performs inner join
  station_with_wesf, left_index=True, right_index=True
).query('WESF > 0').head() # gets the data with positive WESF

# since there were duplicate columns, the left dataframe had x and the right had y appended to them
```

date	PRCP_x	SNOW_x	TMAX	TMIN	TOBS	inclement_weather_x	PRCP_y	SNOW_y	WESF
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=('', '_?') # changes the suffixes
).query('WESF > 0').head()
```

date	PRCP	SNOW	TMAX	TMIN	TOBS	inclement_weather	PRCP_?	SNOW_?	WESF	incl
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() # uses .join() method instead
```

date	PRCP	SNOW	TMAX	TMIN	TOBS	inclement_weather	PRCP_?	SNOW_?	WESF	incl
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) # checks the common stations between weather and station info
# there are 109 as seen in the length parameter
```

```
     Index(['GHCND:US1CTFR0039', 'GHCND:US1NJBG0015', 'GHCND:US1NJBG0017',
            'GHCND:US1NJBG0018', 'GHCND:US1NJBG0023', 'GHCND:US1NJBG0030',
            'GHCND:US1NJBG0039', 'GHCND:US1NJBG0044', 'GHCND:US1NJES0018',
            'GHCND:US1NJES0024',
            ...
            'GHCND:US1NJMS0047', 'GHCND:US1NYSF0083', 'GHCND:US1NYNY0074',
            'GHCND:US1NJPS0018', 'GHCND:US1NJBG0037', 'GHCND:USC00284987',
            'GHCND:US1NJES0031', 'GHCND:US1NJMD0086', 'GHCND:US1NJMS0097',
            'GHCND:US1NJMN0081'],
           dtype='object', length=109)
```

```
weather.index.difference(station_info.index) # weather won't lose rows
```

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

```
station_info.index.difference(weather.index) #station info would lose 153 rows
```

```
     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')] # gets the stations in 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] # checks if formula is same for symmetric difference
```

```
     True
```

```
weather.index.unique().union(station_info.index) # checks the stations that we have after a full outer join
```

```
     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')] # similar to previous code checking for symmetric difference

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
```