```
import pandas as pd

weather = pd.read_csv('/content/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

```
snow_data = weather.query('datatype == "SNOW" and value > 0')
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('/content/weather.db') as connection:
  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
```

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

```
True
```

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

	id	name	latitude	longitude	elevation
0	GHCND:US1CTFR0022	STAMFORD 2.6 SSW, CT US	41.06	-73.58	36.60
1	GHCND:US1CTFR0039	STAMFORD 4.2 S, CT US	41.04	-73.57	6.40
2	GHCND:US1NJBG0001	BERGENFIELD 0.3 SW, NJ US	40.92	-74.00	20.10
3	GHCND:US1NJBG0002	SADDLE BROOK TWP 0.6 E, NJ US	40.90	-74.08	16.80
4	GHCND:US1NJBG0003	TENAFLY 1.3 W, NJ US	40.91	-73.98	21.60

```
weather.head()
```

	attributes	datatype	date	station	value
0	„N,	PRCP	2018-01-01T00:00:00	GHCND:US1CTFR0039	0.00
1	„N,	PRCP	2018-01-01T00:00:00	GHCND:US1NJBG0015	0.00
2	„N,	SNOW	2018-01-01T00:00:00	GHCND:US1NJBG0015	0.00
3	„N,	PRCP	2018-01-01T00:00:00	GHCND:US1NJBG0017	0.00
4	„N,	SNOW	2018-01-01T00:00:00	GHCND:US1NJBG0017	0.00

```
station_info.id.describe()
```

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

```
weather.station.describe()
```

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

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

```
(262, 80256)
```

```
def get_row_count(*dfs):
    return [df.shape[0] for df in dfs]
get_row_count(station_info, weather)
```

```
[262, 80256]
```

```
def get_info(attr, *dfs):
    return list(map(lambda x: getattr(x, attr), dfs))
get_info('shape', station_info, weather)
```

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

```
inner_join = weather.merge(station_info, left_on='station', right_on='id')
inner_join.sample(5, random_state=0)
```

	attributes	datatype	date	station	value	id
27422	,,N,	PRCP	2018-01-23T00:00:00	GHCND:US1NYSF0061	2.30	GHCND:US1NYSF006
19317	T,,N,	PRCP	2018-08-10T00:00:00	GHCND:US1NJUN0014	0.00	GHCND:US1NJUN001
13778	,,N,	WESF	2018-02-18T00:00:00	GHCND:US1NJMS0089	19.60	GHCND:US1NJMS008

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

	attributes	datatype	date	station	value	name	lat
27422	,,N,	PRCP	2018-01-23T00:00:00	GHCND:US1NYSF0061	2.30	CENTERPORT 0.9 SW, NY US	
19317	T,,N,	PRCP	2018-08-10T00:00:00	GHCND:US1NJUN0014	0.00	WESTFIELD 0.6 NE, NJ US	
13778	,,N,	WESF	2018-02-18T00:00:00	GHCND:US1NJMS0089	19.60	PARSIPPANY TROY HILLS TWP 1.3 NJ US	

```
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	id
80404	,,W,	WDF5	2018-12-31T00:00:00	GHCND:USW00094789	130.00	GHCND:USW0009478
80405	,,W,	WSF2	2018-12-31T00:00:00	GHCND:USW00094789	9.80	GHCND:USW0009478

```
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
```

```
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-42-bc483c318509>: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	i
17259	„N,	PRCP	2018-05-15T00:00:00	GHCND:US1NJPS0022	0.30	Na
76178	„N,	PRCP	2018-05-19T00:00:00	GHCND:US1NJPS0015	8.10	Na
73410	„N,	MDPR	2018-08-05T00:00:00	GHCND:US1NYNS0018	12.20	GHCND:US1NYNS001
			2018-04-			

```
import sqlite3

with sqlite3.connect('/content/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('/content/dirty_data2.csv', index_col='date'
).drop_duplicates().drop(columns='SNWD')
dirty_data.head()
```

	station	PRCP	SNOW	TMAX	TMIN	TOBS	WESF	inclement_we
date								
2018-01-01T00:00:00	?	0.00	0.00	5505.00	-40.00	NaN	NaN	
2018-01-02T00:00:00	GHCND:USC00280907	0.00	0.00	-8.30	-16.10	-12.20	NaN	
2018-01-03T00:00:00	GHCND:USC00280907	0.00	0.00	-4.40	-13.90	-13.30	NaN	

```
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()
```

date	PRCP_x	SNOW_x	TMAX	TMIN	TOBS	inclement_weather_x	PRCP_y	SNOW_y	WESF
2018-01-30T00:00:00	0.00	0.00	6.70	-1.70	-0.60	False	1.50	13.00	1.80
2018-03-08T00:00:00	48.80	NaN	1.10	-0.60	1.10	False	28.40	NaN	28.70
2018-03-13T00:00:00	4.10	51.00	5.60	-3.90	0.00	True	3.00	13.00	3.00

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

date	PRCP	SNOW	TMAX	TMIN	TOBS	inclement_weather	PRCP_?	SNOW_?	WESF	ir
2018-01-30T00:00:00	0.00	0.00	6.70	-1.70	-0.60	False	1.50	13.00	1.80	
2018-03-08T00:00:00	48.80	NaN	1.10	-0.60	1.10	False	28.40	NaN	28.70	
2018-03-13T00:00:00	4.10	51.00	5.60	-3.90	0.00	True	3.00	13.00	3.00	

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

date	PRCP	SNOW	TMAX	TMIN	TOBS	inclement_weather	PRCP_?	SNOW_?	WESF	ir
2018-01-30T00:00:00	0.00	0.00	6.70	-1.70	-0.60	False	1.50	13.00	1.80	
2018-03-08T00:00:00	48.80	NaN	1.10	-0.60	1.10	False	28.40	NaN	28.70	
2018-03-13T00:00:00	4.10	51.00	5.60	-3.90	0.00	True	3.00	13.00	3.00	

```
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: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)
```

```
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
```

Comments

Diving into database-style operations on DataFrames with pandas, this document emphasizes the critical skill of manipulating and combining datasets. Querying and merging are foundational techniques in data preprocessing, enabling analysts to filter relevant data and integrate multiple data sources. The weather dataset serves as an excellent example to illustrate these operations, demonstrating how to handle real-world data complexities and prepare datasets for analysis.

Conclusion

In this module it elaborates on various merging strategies, akin to SQL operations, showcasing how different types of joins can be performed with pandas. This knowledge is indispensable in scenarios where data is dispersed across multiple tables or files, requiring a cohesive view. For instance, merging weather data with station information not only enriches the dataset but also illustrates the necessity of combining information from multiple sources to derive comprehensive insights.