# Intro to Data Analysis with Python, Session 2

In this session, we'll discuss interacting with tabular or spreadsheet-like data with the `pandas` package. `pandas` is a high-level library designed to simplify complex operations, as such, it is a very powerful tool well worth learning.

## Interacting with dataframes

`pandas` utilizes a custom data structure called a dataframe. Essentially, dataframes represent tables and contain additional functionality beyond the standard Python data types.

We can load tabular data from a CSV file directly using the `pandas` package. We'll first load some Bluebikes system data (https://www.bluebikes.com/system-data) into a dataframe:

In [ ]:
```
# !pip install pandas  # omitted because pandas is already installed

import pandas as pd

bikes_df = pd.read_csv("shared_data/202303-bluebikes-tripdata.csv")
stations_df = pd.read_csv("shared_data/current_bluebikes_stations.csv")
```

In [ ]:
```
bikes_df.head(10)  #showing top 10 rows
```

Out[ ]:

| | tripduration | starttime | stoptime | start station id | start station name | start station latitude | start station longitude | sta |
|---|---|---|---|---|---|---|---|---|
| **0** | 1105 | 2023-03-01 00:00:44.1520 | 2023-03-01 00:19:09.9080 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **1** | 415 | 2023-03-01 00:01:45.6530 | 2023-03-01 00:08:41.5960 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **2** | 169 | 2023-03-01 00:03:54.2260 | 2023-03-01 00:06:43.6980 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **3** | 624 | 2023-03-01 00:04:13.8340 | 2023-03-01 00:14:38.0830 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **4** | 1116 | 2023-03-01 00:05:04.3640 | 2023-03-01 00:23:40.5370 | 554 | Forsyth St at Huntington Ave | 42.339202 | -71.090511 | |
| **5** | 485 | 2023-03-01 00:05:19.3490 | 2023-03-01 00:13:24.9100 | 141 | Kendall Street | 42.363560 | -71.082168 | |
| **6** | 372 | 2023-03-01 00:05:42.5770 | 2023-03-01 00:11:54.8490 | 554 | Forsyth St at Huntington Ave | 42.339202 | -71.090511 | |
| **7** | 310 | 2023-03-01 00:07:52.0290 | 2023-03-01 00:13:02.6280 | 361 | Deerfield St at Commonwealth Ave | 42.349244 | -71.097282 | |
| **8** | 39772 | 2023-03-01 00:08:46.8850 | 2023-03-01 11:11:39.7260 | 140 | Danehy Park | 42.388966 | -71.132788 | |
| **9** | 268 | 2023-03-01 00:09:16.6710 | 2023-03-01 00:13:44.8920 | 75 | Lafayette Square at Mass Ave / Main St / Colum... | 42.363465 | -71.100573 | |

In [ ]:
```
stations_df.head(10)
```

Out[ ]:

| | Number | Name | Latitude | Longitude | District | Public | Total docks | Deployment Year |
|---|---|---|---|---|---|---|---|---|
| **0** | K32015 | 1200 Beacon St | 42.344149 | -71.114674 | Brookline | Yes | 15 | 2021.0 |
| **1** | W32006 | 160 Arsenal | 42.364664 | -71.175694 | Watertown | Yes | 11 | 2021.0 |
| **2** | A32019 | 175 N Harvard St | 42.363796 | -71.129164 | Boston | Yes | 17 | 2014.0 |
| **3** | S32035 | 191 Beacon St | 42.380323 | -71.108786 | Somerville | Yes | 19 | 2018.0 |
| **4** | C32094 | 2 Hummingbird Lane at Olmsted Green | 42.288870 | -71.095003 | Boston | Yes | 17 | 2020.0 |
| **5** | S32023 | 30 Dane St | 42.381001 | -71.104025 | Somerville | Yes | 15 | 2018.0 |
| **6** | M32026 | 359 Broadway - Broadway at Fayette Street | 42.370803 | -71.104412 | Cambridge | Yes | 23 | 2013.0 |
| **7** | S32049 | 515 Somerville Ave (Temp. Winter Location) | 42.383227 | -71.106069 | NaN | Yes | 19 | NaN |
| **8** | C32106 | 555 Metropolitan Ave | 42.268100 | -71.119240 | Boston | Yes | 18 | 2021.0 |
| **9** | C32105 | 606 American Legion Hwy at Canterbury St | 42.285780 | -71.109725 | Boston | Yes | 18 | 2021.0 |

## Ordered indexing (slicing)

Slicing a dataframe like a list provides succinct access to rows:

In [ ]:
```
# dataframes support slicing, like lists
stations_df[:10]  # note that dataframes are displayed nicely in notebooks witho
```

Out[ ]:

| | Number | Name | Latitude | Longitude | District | Public | Total docks | Deployment Year |
|---|---|---|---|---|---|---|---|---|
| **0** | K32015 | 1200 Beacon St | 42.344149 | -71.114674 | Brookline | Yes | 15 | 2021.0 |
| **1** | W32006 | 160 Arsenal | 42.364664 | -71.175694 | Watertown | Yes | 11 | 2021.0 |
| **2** | A32019 | 175 N Harvard St | 42.363796 | -71.129164 | Boston | Yes | 17 | 2014.0 |
| **3** | S32035 | 191 Beacon St | 42.380323 | -71.108786 | Somerville | Yes | 19 | 2018.0 |
| **4** | C32094 | 2 Hummingbird Lane at Olmsted Green | 42.288870 | -71.095003 | Boston | Yes | 17 | 2020.0 |
| **5** | S32023 | 30 Dane St | 42.381001 | -71.104025 | Somerville | Yes | 15 | 2018.0 |
| **6** | M32026 | 359 Broadway - Broadway at Fayette Street | 42.370803 | -71.104412 | Cambridge | Yes | 23 | 2013.0 |
| **7** | S32049 | 515 Somerville Ave (Temp. Winter Location) | 42.383227 | -71.106069 | NaN | Yes | 19 | NaN |
| **8** | C32106 | 555 Metropolitan Ave | 42.268100 | -71.119240 | Boston | Yes | 18 | 2021.0 |
| **9** | C32105 | 606 American Legion Hwy at Canterbury St | 42.285780 | -71.109725 | Boston | Yes | 18 | 2021.0 |

Elements and sections of dataframes can be accessed easily:

In [ ]:
```
# an element can be accessed using the row and column indices
print(stations_df.iloc[0, 1])
```

```
1200 Beacon St
```

## Reading columns

In [ ]:
```
stations_df.columns
```

Out[ ]:
```
Index(['Number', 'Name', 'Latitude', 'Longitude', 'District', 'Public',
       'Total docks', 'Deployment Year'],
      dtype='object')
```

Dataframes can also be filtered to specific sets of columns:

In [ ]:
```
stations_df[["District", "Deployment Year"]]
```

Out[ ]:

| | District | Deployment Year |
|---|---|---|
| **0** | Brookline | 2021.0 |
| **1** | Watertown | 2021.0 |
| **2** | Boston | 2014.0 |
| **3** | Somerville | 2018.0 |
| **4** | Boston | 2020.0 |
| **...** | ... | ... |
| **443** | Newton | 2020.0 |
| **444** | Boston | 2019.0 |
| **445** | Boston | 2019.0 |
| **446** | Boston | 2018.0 |
| **447** | Somerville | 2012.0 |

448 rows × 2 columns

In [ ]:
```
data_head = stations_df[["District", "Deployment Year"]].head(10)
data_head
```

Out[ ]:

| | District | Deployment Year |
|---|---|---|
| **0** | Brookline | 2021.0 |
| **1** | Watertown | 2021.0 |
| **2** | Boston | 2014.0 |
| **3** | Somerville | 2018.0 |
| **4** | Boston | 2020.0 |
| **5** | Somerville | 2018.0 |
| **6** | Cambridge | 2013.0 |
| **7** | NaN | NaN |
| **8** | Boston | 2021.0 |
| **9** | Boston | 2021.0 |

# Handling missing values

Missing values in datasets are extremely common. You will most certainly come across records where fields have been left empty, and this can cause some big problems for any downstream processing tasks. Empty values can show up as `None` or `NaN` (Not a Number). Some useful tools in dealing with nulls or NAs are provided by `pandas`: `isnull()`, `notnull()`, `dropna()`, and `fillna()`.

## Generating boolean masks

`isnull()` creates a Boolean mask from a sequence indicating which values are NAs:

In [ ]:
```
print(data_head.isnull())
```

```
   District  Deployment Year
0     False            False
1     False            False
2     False            False
3     False            False
4     False            False
5     False            False
6     False            False
7      True             True
8     False            False
9     False            False
```

The opposite mask can be generated with `notnull()`:

In [ ]:
```
print(data_head.notnull())
```

```
   District  Deployment Year
0      True             True
1      True             True
2      True             True
3      True             True
4      True             True
5      True             True
6      True             True
7     False            False
8      True             True
9      True             True
```

## Dropping missing values

`dropna()` simply drops records containing null values:

In [ ]:
```
data_head.dropna()
```

Out[ ]:

| | District | Deployment Year |
|---|---|---|
| **0** | Brookline | 2021.0 |
| **1** | Watertown | 2021.0 |
| **2** | Boston | 2014.0 |
| **3** | Somerville | 2018.0 |
| **4** | Boston | 2020.0 |
| **5** | Somerville | 2018.0 |
| **6** | Cambridge | 2013.0 |
| **8** | Boston | 2021.0 |
| **9** | Boston | 2021.0 |

For a dataframe, dropping is are more complex: we might choose to drop the column or the row, or both. By default, `dropna()` will drop both the row and the column

containing the null value.

In [ ]:
```
df = pd.DataFrame([[1, None, 2],
                   [2, None, 5],
                   [None, 4, 6]],
                  columns=['c1','c2','c3'])

df
```

Out[ ]:

| | c1 | c2 | c3 |
|---|---|---|---|
| **0** | 1.0 | NaN | 2 |
| **1** | 2.0 | NaN | 5 |
| **2** | NaN | 4.0 | 6 |

In [ ]:
```
df.dropna()
```

Out[ ]:

| | c1 | c2 | c3 |
|---|---|---|---|

In [ ]:
```
df.dropna(axis="columns")
```

Out[ ]:

| | c3 |
|---|---|
| **0** | 2 |
| **1** | 5 |
| **2** | 6 |

In [ ]:
```
df.dropna(axis="rows")
```

Out[ ]:

| | c1 | c2 | c3 |
|---|---|---|---|

## Replacing missing values

One alternative to dropping records is to use `fillna()` to fill in the NA values with a specific value:

In [ ]:
```
df.fillna(0)
```

Out[ ]:

| | c1 | c2 | c3 |
|---|---|---|---|
| **0** | 1.0 | 0.0 | 2 |
| **1** | 2.0 | 0.0 | 5 |
| **2** | 0.0 | 4.0 | 6 |

## Concatenating dataframes

Dataframes can be concatenated in a relatively simple way by using the `concat()` function:

In [ ]:
```
x = pd.DataFrame([["A", "B"], ["C", "D"], ["G", "H"]])
y = pd.DataFrame([["E", "F"], ["G", "H"], ["C", "D"]])
```

In [ ]:
```
x
```

Out[ ]:

| | 0 | 1 |
|---|---|---|
| **0** | A | B |
| **1** | C | D |
| **2** | G | H |

In [ ]:
```
y
```

Out[ ]:

| | 0 | 1 |
|---|---|---|
| **0** | E | F |
| **1** | G | H |
| **2** | C | D |

In [ ]:
```
z = pd.concat([x, y])
z
```

Out[ ]:

| | 0 | 1 |
|---|---|---|
| **0** | A | B |
| **1** | C | D |
| **2** | G | H |
| **0** | E | F |
| **1** | G | H |
| **2** | C | D |

## Dropping duplicates

If we want to drop duplicates, we need to specify which occurrence to keep (either first or last):

In [ ]:
```
z1 = z.drop_duplicates(keep="first")
z1
```

Out[ ]:

| | 0 | 1 |
|---|---|---|
| **0** | A | B |
| **1** | C | D |
| **2** | G | H |
| **0** | E | F |

In [ ]:
```
z1.reset_index(drop=True)
```

Out[ ]:

| | 0 | 1 |
|---|---|---|
| **0** | A | B |
| **1** | C | D |
| **2** | G | H |
| **3** | E | F |

# Manipulating tabular data

With `pandas`, we can manipulate tabular data at a fairly advanced level. In this section, we'll cover some of these manipulation operations.

## Filtering

In [ ]:
```
bikes_df.columns
```

Out[ ]:
```
Index(['tripduration', 'starttime', 'stoptime', 'start station id',
       'start station name', 'start station latitude',
       'start station longitude', 'end station id', 'end station name',
       'end station latitude', 'end station longitude', 'bikeid', 'usertype',
       'postal code'],
      dtype='object')
```

In [ ]:
```
bikes_df.head(10)
```

Out[ ]:

| | tripduration | starttime | stoptime | start station id | start station name | start station latitude | start station longitude | sta |
|---|---|---|---|---|---|---|---|---|
| **0** | 1105 | 2023-03-01 00:00:44.1520 | 2023-03-01 00:19:09.9080 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **1** | 415 | 2023-03-01 00:01:45.6530 | 2023-03-01 00:08:41.5960 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **2** | 169 | 2023-03-01 00:03:54.2260 | 2023-03-01 00:06:43.6980 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **3** | 624 | 2023-03-01 00:04:13.8340 | 2023-03-01 00:14:38.0830 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **4** | 1116 | 2023-03-01 00:05:04.3640 | 2023-03-01 00:23:40.5370 | 554 | Forsyth St at Huntington Ave | 42.339202 | -71.090511 | |
| **5** | 485 | 2023-03-01 00:05:19.3490 | 2023-03-01 00:13:24.9100 | 141 | Kendall Street | 42.363560 | -71.082168 | |
| **6** | 372 | 2023-03-01 00:05:42.5770 | 2023-03-01 00:11:54.8490 | 554 | Forsyth St at Huntington Ave | 42.339202 | -71.090511 | |
| **7** | 310 | 2023-03-01 00:07:52.0290 | 2023-03-01 00:13:02.6280 | 361 | Deerfield St at Commonwealth Ave | 42.349244 | -71.097282 | |
| **8** | 39772 | 2023-03-01 00:08:46.8850 | 2023-03-01 11:11:39.7260 | 140 | Danehy Park | 42.388966 | -71.132788 | |
| **9** | 268 | 2023-03-01 00:09:16.6710 | 2023-03-01 00:13:44.8920 | 75 | Lafayette Square at Mass Ave / Main St / Colum... | 42.363465 | -71.100573 | |

In [ ]:
```
bikes_df["start station name"].unique()  # check unique stations in this column
```

Out[ ]:
```
array(['Sennott Park Broadway at Norfolk Street',
       'Ruggles T Stop - Columbus Ave at Melnea Cass Blvd',
       'Forsyth St at Huntington Ave', 'Kendall Street',
       'Deerfield St at Commonwealth Ave', 'Danehy Park',
       'Lafayette Square at Mass Ave / Main St / Columbia St',
       'Christian Science Plaza - Massachusetts Ave at Westland Ave',
       'Roxbury Crossing T Stop - Columbus Ave at Tremont St',
       'Harvard Stadium: N. Harvard St at Soldiers Field Rd',
       'MIT Pacific St at Purrington St', 'Park Dr at Buswell St',
       'MIT at Mass Ave / Amherst St', 'MIT Vassar St',
       'Harvard Law School at Mass Ave / Jarvis St', '955 Mass Ave',
       'MIT Carleton St at Amherst St',
       'Orient Heights T Stop - Bennington St at Saratoga St',
       'Watermark Seaport - Boston Wharf Rd at Seaport Blvd',
       'South Station - 700 Atlantic Ave',
       'Central Square at Mass Ave / Essex St',
       'Commonwealth Ave at Agganis Way',
       'Kennedy-Longfellow School 158 Spring St',
       'Salem MBTA - Washington at Federal St', 'University Park',
       'Perry Park', 'Galileo Galilei Way at Fulkerson St/Binney St',
       'Harvard Kennedy School at Bennett St / Eliot St',
       'Harvard Square at Mass Ave/ Dunster',
       'Harvard University River Houses at DeWolfe St / Cowperthwaite St',
       'Mass Ave at Hadley/Walden',
       "Packard's Corner - Commonwealth Ave at Brighton Ave",
       'Charles Circle - Charles St at Cambridge St',
       'Wentworth Institute of Technology - Huntington Ave at Vancouver St',
       'B.U. Central - 725 Comm. Ave.', 'One Beacon St',
       'Berkshire Street at Cambridge Street', 'Broadway T Stop',
       'Longwood Ave at Binney St',
       'Cambridge Crossing at North First Street',
       'Copley Square - Dartmouth St at Boylston St', 'Silber Way',
       'Grove Hall Library - 41 Geneva Ave',
       'HMS/HSPH - Avenue Louis Pasteur at Longwood Ave',
       'Seaport Blvd at Sleeper St', 'Cambridge St at Joy St',
       'Harvard St at Greene-Rose Heritage Park',
       'Martha Eliot Health Center', 'Boylston St at Jersey St',
       'Verizon Innovation Hub 10 Ware Street',
       'W Broadway at Dorchester St', 'Nashua Street at Red Auerbach Way',
       'Innovation Lab - 125 Western Ave at Batten Way',
       'Chinatown T Stop', 'Gore Street at Lambert Street',
       'Congress St at Northern Ave',
       'Boylston St at Charles St (Temp Winter Location)',
       'Harvard St and Stedman St', 'Mugar Way at Beacon St',
       'Brigham Circle - Francis St at Huntington Ave',
       'Washington St at Brock St',
       'MIT Stata Center at Vassar St / Main St',
       'Central Sq Post Office / Cambridge City Hall at Mass Ave / Pleasant S
t',
       '700 Commonwealth Ave.', 'Stony Brook T Stop',
       'Raymond Park at Walden St', 'Lansdowne T Stop', 'Third at Binney',
       'Landmark Center - Brookline Ave at Park Dr', 'W Broadway at D St',
       'One Broadway / Kendall Sq at Main St / 3rd St',
       'Harrison Ave at Mullins Way',
       'Government Center - Cambridge St at Court St',
       'Stuart St at Charles St',
       'John Ahern Field at Kennedy-Longfellow School',
       'Brighton Center - Washington St at Cambridge St',
       'Rowes Wharf at Atlantic Ave',
       '359 Broadway - Broadway at Fayette Street',
```

'Cross St at Hanover St', 'Commonwealth Ave at Griggs St',
'Lower Cambridgeport at Magazine St / Riverside Rd',
'Community Path at Cedar Street', 'Fan Pier',
'Commonwealth Ave at Kelton St', 'Alewife MBTA at Steel Place',
'Broadway at Central St', 'Porter Square Station', 'Troy Boston',
'Foss Park', 'Alewife Station at Russell Field',
'Union Square - Somerville', 'Valenti Way at Haverhill St',
'Chelsea St at Vine St', 'Harrison Ave at E. Dedham St',
'St. Alphonsus St at Tremont St', 'Ashmont T Stop',
'Harvard University Radcliffe Quadrangle at Shepard St / Garden St',
'Columbus Ave at W. Canton St',
'South End Library - Tremont St at W Newton St',
'Tremont St at E Berkeley St',
'Child Street at Brian P. Murphy Staircase',
'Linear Park - Mass. Ave. at Cameron Ave. ',
'Seaport Square - Seaport Blvd at Northern Ave',
'Boston East - 126 Border St', 'Tremont St at Northampton St',
'Centre St at W. Roxbury Post Office',
'Hyde Square - Barbara St at Centre St',
'Harvard Ave at Brainerd Rd', 'Charlestown Navy Yard',
'Union Square - Brighton Ave at Cambridge St',
'Somerville City Hall', 'Chinatown Gate Plaza',
'Conway Park - Somerville Avenue', 'Tremont St at W. Dedham St',
'Warren St at Chelsea St', 'Lesley University',
'Union Square East', 'Huntington Ave at Mass Art',
'Aquarium T Stop - 200 Atlantic Ave',
'Thorndike Field at Minuteman Bikeway',
'South Boston Library - 646 E Broadway', 'Washington Sq',
'Dana Park', 'Davis Square', 'Railroad Lot and Minuteman Bikeway',
'Graham and Parks School – Linnaean St at Walker St',
'Boylston St at Massachusetts Ave',
'Sidney Research Campus/Erie Street at Waverly',
'Washington St at Rutland St',
'Roxbury YMCA - Warren St at MLK Blvd', 'Kenmore Square',
'Boston Convention and Exhibition Center - Summer St at West Side Dr',
'Main St at Austin St', 'Edgerly Education Center',
'Old Morse Park at Putnam Ave', 'Boston City Hall - 28 State St',
'Tremont St at Hamilton Pl', "St Mary's",
'Green Street T Stop - Green St at Amory St',
'Commonwealth Ave At Babcock St',
'Broadway Opposite Norwood Ave  (Temp Winter Station)',
'Brookline Village - Station Street at MBTA', 'Honan Library',
'699 Mt Auburn St', 'Harrison Ave at Bennet St',
'Commonwealth Ave at Chiswick Rd',
'Curtis Hall - South St at Centre St',
'Main Street/Albany Street/Technology Square',
'Washington St at Waltham St', '30 Dane St',
'Cambridge St - at Columbia St / Webster Ave',
'Linwood St at Minuteman Bikeway',
'Harvard University / SEAS Cruft-Pierce Halls at 29 Oxford St',
'515 Somerville Ave (Temp. Winter Location)',
'Tufts Sq - Main St at Medford St',
'Cambridge Main Library at Broadway / Trowbridge St',
'Bartlett St at John Elliot Sq', "The Overlook at St. Gabriel's",
'Ink Block - Harrison Ave at Herald St',
'Central Square East Boston', 'Marion St at Harvard St',
'Brighton Mills - 370 Western Ave', 'Lewis Wharf at Atlantic Ave',
'Oak Square - 615 Washington St', '191 Beacon St',
'Harvard University Housing - 115 Putnam Ave at Peabody Terrace',
'Packard Ave at Powderhouse Blvd', '175 N Harvard St',

'Surface Rd at Summer St',
'Edwards Playground - Main St at Eden St',
'Broadway St at Mt Pleasant St',
'Washington St @ New Washington St (Temp Winter Station)',
'Mass Ave T Station', 'Seaport Hotel - Congress St at Seaport Ln',
'1200 Beacon St', 'Inman Square at Springfield St.',
'One Kendall Square at Hampshire St / Portland St',
'MIT Hayward St at Amherst St',
'Columbia Rd at Tierney Community Center',
'Back Bay T Stop - Dartmouth St at Stuart St',
'Ames St at Main St', 'Faneuil St at Arlington St',
'Coolidge Corner - Beacon St at Centre St', 'Ames St at Broadway',
'Main St at Baldwin St', 'State Street at Channel Center',
'Forest Hills', 'Harvard Square at Brattle St / Eliot St',
'Colleges of the Fenway - Fenway at Avenue Louis Pasteur',
'Beacon St at Tappan St', '855 Broadway', 'Main St at Thompson Sq',
'Discovery Park - 30 Acorn Park Drive',
'East Somerville Library (Broadway and Illinois)',
'Conway Park @ Bleachery Ct (Temp Winter Station)',
'Purchase St at Pearl St', 'The Dimock Center',
'Albany St at E. Brookline St',
'Powder House Circle - Nathan Tufts Park', 'One Brigham Circle',
'Shawmut Ave at Oak St W', 'Surface Rd at India St',
'Farragut Rd at E. 6th St', 'ID Building East',
'Medford St at Charlestown BCYF', 'Cypress St at Clark Playground',
'Brookline Town Hall', 'Flat 9 at Whittier', 'Kendall T',
'Cambridge Dept. of Public Works -147 Hampshire St.',
'MLK Blvd at Washington St', 'Boylston St at Fairfield St',
'Cleveland Circle', 'Bowdoin St at Quincy St',
'Broad Canal Way at Third Street',
'JFK Crossing at Harvard St. / Thorndike St.',
'84 Cambridgepark Dr', 'Huron Ave At Vassal Lane',
'Tremont St at West St', 'Mass Ave at Albany St', '700 Huron Ave',
'Shawmut Ave at Lenox St', 'Dudley Square - Bolling Building',
'Washington St at Lenox St', 'West End Park', '75 Binney St',
'Grove St at Community Path',
'CambridgeSide Galleria - CambridgeSide PL at Land Blvd',
'Dorchester Ave at Gillette Park', 'Boylston St at Berkeley St',
'Gove St at Orleans St', 'Prudential Center - 101 Huntington Ave',
'Williams St at Washington St', 'Stuart St at Berkeley St',
'Washington St at Melnea Cass Blvd',
"Rindge Avenue - O'Neill Library",
'Soldiers Field Park - 111 Western Ave', 'Museum of Science',
'Binney St / Sixth St', 'Hyde Park Ave at Walk Hill St',
'Assembly Square T', 'Sydney St at Carson St',
'Jackson Square T Stop', 'Gilman Square T at Medford St',
'Roslindale Village - Washington St', 'Arsenal Yards',
'Gramsdorf Playground', 'High St at Cypress St',
'Spaulding Rehabilitation Hospital - Charlestown Navy Yard',
'Teele Square', 'Somerville High School & Central Library',
'Rogers St & Land Blvd', 'Wilson Square', 'EF - North Point Park',
'Bunker Hill Community College',
'Galileo Galilei Way at Main Street', 'One Memorial Drive',
'Mt Auburn', 'Northern Strand at Main St', 'Shawmut T Stop',
'Tappan St at Brookline Hills MBTA', 'Hood Park',
'Danehy Park at New Street',
'Harvard University Gund Hall at Quincy St / Kirkland St',
'Magoun Square at Trum Field', 'Pier 4 Blvd at Autumn Ln',
'Sullivan Square', 'New Balance - 20 Guest St',
'Roseland St At Dorchester Ave', 'Codman Square Library',

'Walnut Ave at School St', 'The Lawn on D',
'Four Corners - 157 Washington St', 'Jamaica St at South St',
'Congress St at Boston City Hall', 'Park Plaza at Charles St S.',
'Newmarket Square T Stop - Massachusetts Ave at Newmarket Square',
'Washington St at Talbot Ave', 'Day Sq',
'Concord Ave at Spinelli Place', 'Sumner St at Shirley Ave',
'Savin Hill T Stop - S Sydney St at Bay St',
'Maverick Square - Lewis Mall', 'Malden High School',
'Murphy Skating Rink - 1880 Day Blvd',
'Trum Field @ Cedar St (Temp Winter Station)',
'Medford Sq - Riverside Ave at River St',
'Boylston St at Exeter St', 'ID Building West',
'Clinton St at North St', 'Mayor Salvo Path at Mill St',
'Smith Pl at Wilson Rd', 'Encore',
'Geiger Gibson Community Health Center', 'Spring St at Powell St',
'East Boston Neighborhood Health Center - 20 Maverick Square',
'Chelsea Station', 'Nichols Ave. at Watertown Greenway',
'Columbia Rd at Ceylon St', 'Whittier St Health Center',
'Franklin Park Zoo - Franklin Park Rd at Blue Hill Ave',
'S Huntington Ave at Heath St',
'Boynton Yards at 101 South Street', 'Main St at Brooks Park',
'Porzio Park', 'Franklin Park - Seaver St at Humboldt Ave',
'Gallivan Blvd at Adams St', 'Foley St at Grand Union Blvd',
'JFK/UMass T Stop', 'Roslindale Village - South St',
'Orr Sq (Shirley Ave)',
'Salem State University - Bike Path at Loring Ave', 'Coolidge Sq.',
'Community Path at Lowell St',
'Glendale Square (Ferry St at Broadway)',
'Egleston Square - Atherton St at Washington St', 'Boston Landing',
'Malden Center T Station', '645 Summer St',
'Centre St at Parkway YMCA', 'NCAAA - Walnut Ave at Crawford St',
'87-101 Cambridgepark Drive', 'Wasgott Playground', 'Watertown Sq',
'7 Acre Park', 'Shetland Park - Congress at Peabody St',
'160 Arsenal',
'University of Massachusetts Boston - Campus Center',
'Everett Square (Broadway at Chelsea St)',
'Salem MBTA - Lower Level', 'Broadway at Lynde St',
'Revere City Hall', 'Bellingham Square',
'Vassal Lane at Tobin/VLUS', 'Belgrade Ave at Walworth St',
'Bennington St at Constitution Beach', 'Broadway at Maple St',
'Lafayette at Leach St', 'Fresh Pond Reservation',
'Hawthorne Boulevard', 'Kearins Playground',
'The Eddy - New St at Sumner St', 'Cleary Sq',
'Maverick St at Massport Path', 'Chelsea Square',
'Circuit Drive at American Legion Hwy',
'Bennington St at Byron St',
'Airport T Stop - Bremen St at Brooks St', 'Watertown Town Hall',
'North St at Liberty Hill Ave', 'Broadway at Beacham St',
'Blue Hill Ave at Havelock St',
'Salem State University - North Campus', 'Wellington MBTA',
'Mattapan T Stop', '555 Metropolitan Ave',
'Morton St at Gallivan Blvd', 'Ross Playground',
'Talbot Ave At Blue Hill Ave',
'606 American Legion Hwy at Canterbury St', 'Swan St. Park',
'Cary Square', 'Goodhue St at Grove St', 'Broadway at Gerrish Ave',
'Washington St at Myrtle St', 'Clarendon Hill at Broadway',
'Revere Public Library', 'Essex St at Dalton Parkway',
'Beacon St at Washington / Kirkland', 'Craigie at Summer St',
'American Legion Hwy at Cummins Hwy', 'Forest River Park',
'Hyde Park Library', 'Mattapan Library', 'Elm St at White St',

```
'Somerville Hospital', 'Somerville City Hall Annex', 'Piers Park',
'Ball Sq', 'Marion St at White St',
'Burlington Ave at Brookline Ave', 'Glendon St at Condor St',
'Western Ave at Richardson St', 'Addison St at Saratoga St',
'Washington St at Egremont Rd', 'Blue Hill Ave at Almont St',
'Canal St at Causeway St', 'Blossom St at Charles St',
'Broad St at Central St',
'Boston Medical Center - E Concord St at Harrison Ave',
'Arch St at Franklin St', 'Uphams Corner',
'Uphams Corner T Stop - Magnolia St at Dudley St',
'2 Hummingbird Lane at Olmsted Green', 'Walnut Ave at Warren St',
'Riverway at Brookline Ave', 'E Cottage St at Columbia Rd',
'Community Life Center',
'Dudley Town Common - Mt Pleasant Ave at Blue Hill Ave',
'Fields Corner T Stop', 'Northbourne Rd at Hyde Park Ave',
'Archdale Rd at Washington St', 'Mt. Hope St at Hyde Park Ave',
'Thetford Ave at Norfolk St'], dtype=object)
```

Say we only want data for the trips that started at University Park. We can create a boolean mask to accomplish this. A boolean mask is simply an ordered array of `True` / `False` values indicating which elements to choose.

In [ ]:
```
bikes_df["start station name"] == "University Park"
```

Out[ ]:
```
0         False
1         False
2         False
3         False
4         False
          ...
198998    False
198999    False
199000    False
199001    False
199002    False
Name: start station name, Length: 199003, dtype: bool
```

In [ ]:
```
bikes_df[bikes_df["start station name"] == "University Park"]
```

Out[ ]:

| | tripduration | starttime | stoptime | start station id | start station name | start station latitude | start station longitude | sta |
|---|---|---|---|---|---|---|---|---|
| **36** | 311 | 2023-03-01 00:34:06.9730 | 2023-03-01 00:39:18.4960 | 177 | University Park | 42.362648 | -71.100061 | |
| **67** | 911 | 2023-03-01 01:07:21.7210 | 2023-03-01 01:22:32.9700 | 177 | University Park | 42.362648 | -71.100061 | |
| **282** | 1045 | 2023-03-01 08:02:23.9960 | 2023-03-01 08:19:49.7660 | 177 | University Park | 42.362648 | -71.100061 | |
| **353** | 735 | 2023-03-01 08:16:59.3930 | 2023-03-01 08:29:14.9870 | 177 | University Park | 42.362648 | -71.100061 | |
| **359** | 611 | 2023-03-01 08:19:10.0260 | 2023-03-01 08:29:21.1300 | 177 | University Park | 42.362648 | -71.100061 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **198963** | 82 | 2023-03-31 23:26:34.6530 | 2023-03-31 23:27:57.0220 | 177 | University Park | 42.362648 | -71.100061 | |
| **198967** | 978 | 2023-03-31 23:29:43.3400 | 2023-03-31 23:46:02.1560 | 177 | University Park | 42.362648 | -71.100061 | |
| **198972** | 364 | 2023-03-31 23:32:54.4360 | 2023-03-31 23:38:58.7640 | 177 | University Park | 42.362648 | -71.100061 | |
| **198976** | 407 | 2023-03-31 23:37:20.8540 | 2023-03-31 23:44:07.8970 | 177 | University Park | 42.362648 | -71.100061 | |
| **198988** | 635 | 2023-03-31 23:50:31.4760 | 2023-04-01 00:01:06.6540 | 177 | University Park | 42.362648 | -71.100061 | |

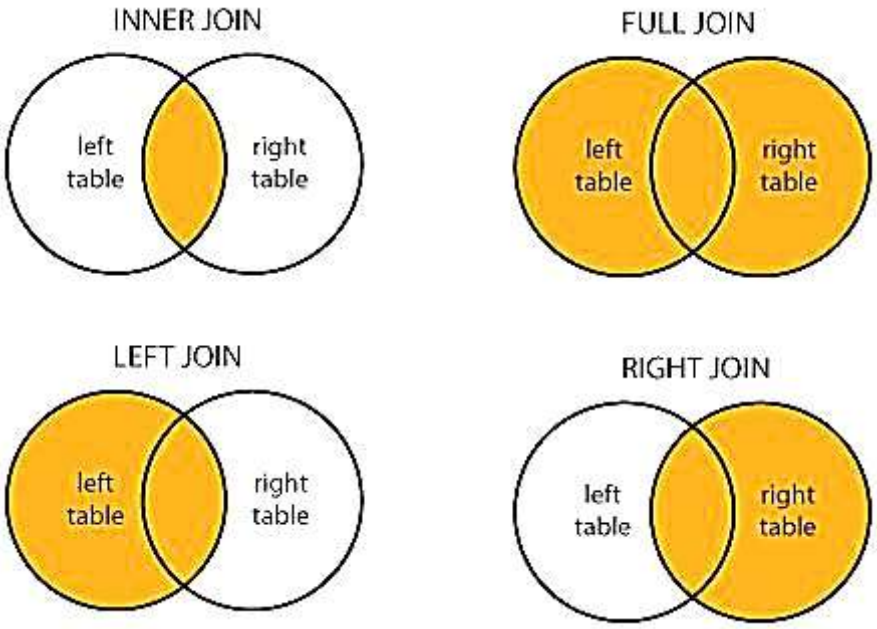
1337 rows × 14 columns

## Grouping and joins

Grouping operations are common in relational database systems such as SQL, but we can also use `pandas` to perform some of them. There are different types of join

operations. These operations are usually performed on two tables, a "left" table and a "right" table.

Joins are often compared with set operations and illustrated with Venn diagrams like this:




Some of the functionality is similar to the "XLOOKUP" or "VLOOKUP" in Excel (but you can do much more in pandas).

However, joins aren't really set operations. They are actually Cartesian product operations, so Venn diagrams don't tell the whole story.

In [ ]:
```
stations_df.head(10)
```

Out[ ]:

| | Number | Name | Latitude | Longitude | District | Public | Total docks | Deployment Year |
|---|---|---|---|---|---|---|---|---|
| **0** | K32015 | 1200 Beacon St | 42.344149 | -71.114674 | Brookline | Yes | 15 | 2021.0 |
| **1** | W32006 | 160 Arsenal | 42.364664 | -71.175694 | Watertown | Yes | 11 | 2021.0 |
| **2** | A32019 | 175 N Harvard St | 42.363796 | -71.129164 | Boston | Yes | 17 | 2014.0 |
| **3** | S32035 | 191 Beacon St | 42.380323 | -71.108786 | Somerville | Yes | 19 | 2018.0 |
| **4** | C32094 | 2 Hummingbird Lane at Olmsted Green | 42.288870 | -71.095003 | Boston | Yes | 17 | 2020.0 |
| **5** | S32023 | 30 Dane St | 42.381001 | -71.104025 | Somerville | Yes | 15 | 2018.0 |
| **6** | M32026 | 359 Broadway - Broadway at Fayette Street | 42.370803 | -71.104412 | Cambridge | Yes | 23 | 2013.0 |
| **7** | S32049 | 515 Somerville Ave (Temp. Winter Location) | 42.383227 | -71.106069 | NaN | Yes | 19 | NaN |
| **8** | C32106 | 555 Metropolitan Ave | 42.268100 | -71.119240 | Boston | Yes | 18 | 2021.0 |
| **9** | C32105 | 606 American Legion Hwy at Canterbury St | 42.285780 | -71.109725 | Boston | Yes | 18 | 2021.0 |

In [ ]:
```
bikes_df.head(10)
```

Out[ ]:

| | tripduration | starttime | stoptime | start station id | start station name | start station latitude | start station longitude | sta |
|---|---|---|---|---|---|---|---|---|
| **0** | 1105 | 2023-03-01 00:00:44.1520 | 2023-03-01 00:19:09.9080 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **1** | 415 | 2023-03-01 00:01:45.6530 | 2023-03-01 00:08:41.5960 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **2** | 169 | 2023-03-01 00:03:54.2260 | 2023-03-01 00:06:43.6980 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **3** | 624 | 2023-03-01 00:04:13.8340 | 2023-03-01 00:14:38.0830 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **4** | 1116 | 2023-03-01 00:05:04.3640 | 2023-03-01 00:23:40.5370 | 554 | Forsyth St at Huntington Ave | 42.339202 | -71.090511 | |
| **5** | 485 | 2023-03-01 00:05:19.3490 | 2023-03-01 00:13:24.9100 | 141 | Kendall Street | 42.363560 | -71.082168 | |
| **6** | 372 | 2023-03-01 00:05:42.5770 | 2023-03-01 00:11:54.8490 | 554 | Forsyth St at Huntington Ave | 42.339202 | -71.090511 | |
| **7** | 310 | 2023-03-01 00:07:52.0290 | 2023-03-01 00:13:02.6280 | 361 | Deerfield St at Commonwealth Ave | 42.349244 | -71.097282 | |
| **8** | 39772 | 2023-03-01 00:08:46.8850 | 2023-03-01 11:11:39.7260 | 140 | Danehy Park | 42.388966 | -71.132788 | |
| **9** | 268 | 2023-03-01 00:09:16.6710 | 2023-03-01 00:13:44.8920 | 75 | Lafayette Square at Mass Ave / Main St / Colum... | 42.363465 | -71.100573 | |

## Joins

There are different kinds of joins shown in the slide. Here we will see the most common type of join, the left join.

- `left` the left table to be joined
- `right` the right table to be joined
- `how` the type of join to be performed; one of: `'inner'`, `'left'`, `'right'`, or `'outer'`
- `'on'` the column name to be used as keys in both tables

- `left_on` the column name to be used on `left`, only
- `right_on` the column name to be used on `right`, only
- `left_index` If True, use the index (row labels) from the left table as its join key(s).
- `right_index` If True, use the index (row labels) from the right table as its join key(s).

In [ ]:
```
join_df = pd.merge(
    bikes_df,
    stations_df,
    how="left",
    left_on="start station name",
    right_on="Name",
)
join_df
```

Out[ ]:

| | tripduration | starttime | stoptime | start station id | start station name | start station latitude | start station longitude | s |
|---|---|---|---|---|---|---|---|---|
| **0** | 1105 | 2023-03-01 00:00:44.1520 | 2023-03-01 00:19:09.9080 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **1** | 415 | 2023-03-01 00:01:45.6530 | 2023-03-01 00:08:41.5960 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **2** | 169 | 2023-03-01 00:03:54.2260 | 2023-03-01 00:06:43.6980 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **3** | 624 | 2023-03-01 00:04:13.8340 | 2023-03-01 00:14:38.0830 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **4** | 1116 | 2023-03-01 00:05:04.3640 | 2023-03-01 00:23:40.5370 | 554 | Forsyth St at Huntington Ave | 42.339202 | -71.090511 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **198998** | 322 | 2023-03-31 23:55:01.5580 | 2023-04-01 00:00:23.8260 | 550 | Somerville High School & Central Library | 42.386400 | -71.096010 | |
| **198999** | 748 | 2023-03-31 23:55:24.1380 | 2023-04-01 00:07:52.3050 | 61 | Boylston St at Fairfield St | 42.348804 | -71.082369 | |
| **199000** | 1071 | 2023-03-31 23:58:04.4970 | 2023-04-01 00:15:56.0240 | 381 | Inman Square at Springfield St. | 42.374267 | -71.100265 | |
| **199001** | 1096 | 2023-03-31 23:58:37.6520 | 2023-04-01 00:16:53.7120 | 4 | Tremont St at E Berkeley St | 42.345392 | -71.069616 | |
| **199002** | 747 | 2023-03-31 23:59:20.7570 | 2023-04-01 00:11:48.3580 | 515 | 955 Mass Ave | 42.368952 | -71.109988 | |

199003 rows × 22 columns

In [ ]:
```
join_df.columns
```

Out[ ]:
```
Index(['tripduration', 'starttime', 'stoptime', 'start station id',
       'start station name', 'start station latitude',
       'start station longitude', 'end station id', 'end station name',
       'end station latitude', 'end station longitude', 'bikeid', 'usertype',
       'postal code', 'Number', 'Name', 'Latitude', 'Longitude', 'District',
       'Public', 'Total docks', 'Deployment Year'],
      dtype='object')
```

We can remove the redundant columns, if desired:

In [ ]:
```
join_df = join_df[
    [
        "tripduration",
        "starttime",
        "stoptime",
        "start station id",
        "start station name",
        "start station latitude",
        "start station longitude",
        "end station id",
        "end station name",
        "end station latitude",
        "end station longitude",
        "bikeid",
        "usertype",
        "postal code",
        "Number",
        "Name",
        "District",
        "Public",
        "Total docks",
        "Deployment Year",
    ]
]
```

Now we will rename the new columns:

In [ ]:
```
join_df_edited = join_df.rename(
    {
        "Number": "start st number",
        "Name": "start st name",
        "District": "start st district",
        "Public": "start st public",
        "Total docks": "start st total docks",
        "Deployment Year": "start st deployment year",
    },
    axis=1,
)
```

We will again join, but this time based on the end station:

In [ ]:
```
join_df_final = pd.merge(
    join_df_edited,
    stations_df,
    how="left",
```

```
    left_on="end station name",
    right_on="Name",
)
```

In [ ]:
```
join_df_final
```

Out[ ]:

| | tripduration | starttime | stoptime | start station id | start station name | start station latitude | start station longitude | s |
|---|---|---|---|---|---|---|---|---|
| **0** | 1105 | 2023-03-01 00:00:44.1520 | 2023-03-01 00:19:09.9080 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **1** | 415 | 2023-03-01 00:01:45.6530 | 2023-03-01 00:08:41.5960 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **2** | 169 | 2023-03-01 00:03:54.2260 | 2023-03-01 00:06:43.6980 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **3** | 624 | 2023-03-01 00:04:13.8340 | 2023-03-01 00:14:38.0830 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **4** | 1116 | 2023-03-01 00:05:04.3640 | 2023-03-01 00:23:40.5370 | 554 | Forsyth St at Huntington Ave | 42.339202 | -71.090511 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **198998** | 322 | 2023-03-31 23:55:01.5580 | 2023-04-01 00:00:23.8260 | 550 | Somerville High School & Central Library | 42.386400 | -71.096010 | |
| **198999** | 748 | 2023-03-31 23:55:24.1380 | 2023-04-01 00:07:52.3050 | 61 | Boylston St at Fairfield St | 42.348804 | -71.082369 | |
| **199000** | 1071 | 2023-03-31 23:58:04.4970 | 2023-04-01 00:15:56.0240 | 381 | Inman Square at Springfield St. | 42.374267 | -71.100265 | |
| **199001** | 1096 | 2023-03-31 23:58:37.6520 | 2023-04-01 00:16:53.7120 | 4 | Tremont St at E Berkeley St | 42.345392 | -71.069616 | |
| **199002** | 747 | 2023-03-31 23:59:20.7570 | 2023-04-01 00:11:48.3580 | 515 | 955 Mass Ave | 42.368952 | -71.109988 | |

199003 rows × 28 columns

In [ ]:
```
join_df_final.columns
```

Out[ ]:
```
Index(['tripduration', 'starttime', 'stoptime', 'start station id',
       'start station name', 'start station latitude',
       'start station longitude', 'end station id', 'end station name',
       'end station latitude', 'end station longitude', 'bikeid', 'usertype',
       'postal code', 'start st number', 'start st name', 'start st district',
       'start st public', 'start st total docks', 'start st deployment year',
       'Number', 'Name', 'Latitude', 'Longitude', 'District', 'Public',
       'Total docks', 'Deployment Year'],
      dtype='object')
```

In [ ]:
```
join_df_final = join_df_final[
    [
        "tripduration",
        "starttime",
        "stoptime",
        "start station id",
        "start station name",
        "start station latitude",
        "start station longitude",
        "end station id",
        "end station name",
        "end station latitude",
        "end station longitude",
        "bikeid",
        "usertype",
        "postal code",
        "start st number",
        "start st name",
        "start st district",
        "start st public",
        "start st total docks",
        "start st deployment year",
        "Number",
        "Name",
        "District",
        "Public",
        "Total docks",
        "Deployment Year",
    ]
]

join_df_final = join_df_final.rename(
    {
        "Number": "end st number",
        "Name": "end st name",
        "District": "end st district",
        "Public": "end st public",
        "Total docks": "end st total docks",
        "Deployment Year": "end st deployment year",
    },
    axis=1,
)

join_df_final.columns
```

Out[ ]:
```
Index(['tripduration', 'starttime', 'stoptime', 'start station id',
       'start station name', 'start station latitude',
       'start station longitude', 'end station id', 'end station name',
       'end station latitude', 'end station longitude', 'bikeid', 'usertype',
       'postal code', 'start st number', 'start st name', 'start st district',
       'start st public', 'start st total docks', 'start st deployment year',
       'end st number', 'end st name', 'end st district', 'end st public',
       'end st total docks', 'end st deployment year'],
      dtype='object')
```

In [ ]:
```
join_df_final = join_df_final[
    [
        "tripduration",
        "starttime",
        "stoptime",
        "start station id",
        "start station name",
        "start station latitude",
        "start station longitude",
        "end station id",
        "end station name",
        "end station latitude",
        "end station longitude",
        "bikeid",
        "usertype",
        "postal code",
        "start st number",
        "start st district",
        "start st public",
        "start st total docks",
        "start st deployment year",
        "end st number",
        "end st district",
        "end st public",
        "end st total docks",
        "end st deployment year",
    ]
]

join_df_final
```

Out[ ]:

| | tripduration | starttime | stoptime | start station id | start station name | start station latitude | start station longitude | s |
|---|---|---|---|---|---|---|---|---|
| **0** | 1105 | 2023-03-01 00:00:44.1520 | 2023-03-01 00:19:09.9080 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **1** | 415 | 2023-03-01 00:01:45.6530 | 2023-03-01 00:08:41.5960 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **2** | 169 | 2023-03-01 00:03:54.2260 | 2023-03-01 00:06:43.6980 | 12 | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | 42.336244 | -71.087986 | |
| **3** | 624 | 2023-03-01 00:04:13.8340 | 2023-03-01 00:14:38.0830 | 386 | Sennott Park Broadway at Norfolk Street | 42.368605 | -71.099302 | |
| **4** | 1116 | 2023-03-01 00:05:04.3640 | 2023-03-01 00:23:40.5370 | 554 | Forsyth St at Huntington Ave | 42.339202 | -71.090511 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **198998** | 322 | 2023-03-31 23:55:01.5580 | 2023-04-01 00:00:23.8260 | 550 | Somerville High School & Central Library | 42.386400 | -71.096010 | |
| **198999** | 748 | 2023-03-31 23:55:24.1380 | 2023-04-01 00:07:52.3050 | 61 | Boylston St at Fairfield St | 42.348804 | -71.082369 | |
| **199000** | 1071 | 2023-03-31 23:58:04.4970 | 2023-04-01 00:15:56.0240 | 381 | Inman Square at Springfield St. | 42.374267 | -71.100265 | |
| **199001** | 1096 | 2023-03-31 23:58:37.6520 | 2023-04-01 00:16:53.7120 | 4 | Tremont St at E Berkeley St | 42.345392 | -71.069616 | |
| **199002** | 747 | 2023-03-31 23:59:20.7570 | 2023-04-01 00:11:48.3580 | 515 | 955 Mass Ave | 42.368952 | -71.109988 | |

199003 rows × 24 columns

## Grouping

We can perform grouping operations in order to aggregate data. The functionality is similar to the "SUMIF"/"SUMIFS"/"COUNTIF" etc. functions in Excel.

Let's say we want to know the total trip duration between each OD pair:

In [ ]:
```
trip_duration_df = join_df_final[
    ["start station name", "end station name", "tripduration"]
]

trip_duration_df
```

Out[ ]:

| | start station name | end station name | tripduration |
|---|---|---|---|
| **0** | Sennott Park Broadway at Norfolk Street | Marion St at Harvard St | 1105 |
| **1** | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | Brigham Circle - Francis St at Huntington Ave | 415 |
| **2** | Ruggles T Stop - Columbus Ave at Melnea Cass Blvd | Wentworth Institute of Technology - Huntington... | 169 |
| **3** | Sennott Park Broadway at Norfolk Street | 191 Beacon St | 624 |
| **4** | Forsyth St at Huntington Ave | Stony Brook T Stop | 1116 |
| **...** | ... | ... | ... |
| **198998** | Somerville High School & Central Library | Somerville Hospital | 322 |
| **198999** | Boylston St at Fairfield St | MIT Pacific St at Purrington St | 748 |
| **199000** | Inman Square at Springfield St. | Copley Square - Dartmouth St at Boylston St | 1071 |
| **199001** | Tremont St at E Berkeley St | Berkshire Street at Cambridge Street | 1096 |
| **199002** | 955 Mass Ave | MIT Carleton St at Amherst St | 747 |

199003 rows × 3 columns

In [ ]:
```
od_trip_duration_df = (
    trip_duration_df.groupby(["start station name", "end station name"])[
        "tripduration"
    ]
    .sum()
    .reset_index()
)

od_trip_duration_df
```

Out[ ]:

| | start station name | end station name | tripduration |
|---|---|---|---|
| **0** | 1200 Beacon St | 1200 Beacon St | 23440 |
| **1** | 1200 Beacon St | 30 Dane St | 1598 |
| **2** | 1200 Beacon St | 359 Broadway - Broadway at Fayette Street | 1727 |
| **3** | 1200 Beacon St | 515 Somerville Ave (Temp. Winter Location) | 1504 |
| **4** | 1200 Beacon St | 699 Mt Auburn St | 4460 |
| **...** | ... | ... | ... |
| **32915** | Wilson Square | Verizon Innovation Hub 10 Ware Street | 789 |
| **32916** | Wilson Square | Warren St at Chelsea St | 1262 |
| **32917** | Wilson Square | Washington St @ New Washington St (Temp Winter... | 712 |
| **32918** | Wilson Square | Wentworth Institute of Technology - Huntington... | 2042 |
| **32919** | Wilson Square | Wilson Square | 14716 |

32920 rows × 3 columns

Now let's say we want to know the total trip duration by start station:

In [ ]:
```
origin_trip_duration_df = (
    trip_duration_df.groupby("start station name")["tripduration"]
    .sum()
    .reset_index()
)

origin_trip_duration_df
```

Out[ ]:

| | start station name | tripduration |
|---|---|---|
| **0** | 1200 Beacon St | 558540 |
| **1** | 160 Arsenal | 108989 |
| **2** | 175 N Harvard St | 729479 |
| **3** | 191 Beacon St | 393446 |
| **4** | 2 Hummingbird Lane at Olmsted Green | 3737 |
| **...** | ... | ... |
| **399** | West End Park | 230260 |
| **400** | Western Ave at Richardson St | 98612 |
| **401** | Whittier St Health Center | 80719 |
| **402** | Williams St at Washington St | 362949 |
| **403** | Wilson Square | 340482 |

404 rows × 2 columns

# Writing data to files

We can use a dataframe's `to_csv()` method to save it to a CSV file:

In [ ]:
```
origin_trip_duration_df.to_csv("origin_trip_duration.csv")
```

Specify `index=False` to omit indices from your output file:

In [ ]:
```
origin_trip_duration_df.to_csv(
    "origin_trip_duration_no_index.csv", index=False
)
```