Submitted by: Angelo Luis C. Cu

```
import numpy as np
import pandas as pd
weather = pd.read_csv('data/weather_by_station.csv', index_col='date', parse_dates=True)
weather.head()
```

date	datatype	station	value	station_name
2018-01-01	PRCP	GHCND:US1CTFR0039	0.0	STAMFORD 4.2 S, CT US
2018-01-01	PRCP	GHCND:US1NJBG0015	0.0	NORTH ARLINGTON 0.7 WNW, NJ US
2018-01-01	SNOW	GHCND:US1NJBG0015	0.0	NORTH ARLINGTON 0.7 WNW, NJ US
2018-01-01	PRCP	GHCND:US1NJBG0017	0.0	GLEN ROCK 0.7 SSE, NJ US
2018-01-01	SNOW	GHCND:US1NJBG0017	0.0	GLEN ROCK 0.7 SSE, NJ US

Next steps: View recommended plots

```
fb = pd.read_csv('data/fb_2018.csv', index_col='date', parse_dates=True).assign(
  trading_volume=lambda x: pd.cut(x.volume, bins=3, labels=['low', 'med', 'high'])
)
fb.head()
```

date	open	high	low	close	volume	trading_volume
2018-01-02	177.68	181.58	177.5500	181.42	18151903	low
2018-01-03	181.88	184.78	181.3300	184.67	16886563	low
2018-01-04	184.90	186.21	184.0996	184.33	13880896	low
2018-01-05	185.59	186.90	184.9300	186.85	13574535	low
2018-01-08	187.20	188.90	186.3300	188.28	17994726	low

Next steps: View recommended plots

```
pd.set_option('display.float_format', lambda x: '%.2f' % x)
# sets floats to be displayed with 2 decimal places instead of scientific notation
```

```
fb.agg({ # performs the operations and stores them in a series
  'open': np.mean,
  'high': np.max,
  'low': np.min,
  'close': np.mean,
  'volume': np.sum
})
```

```
open            171.45
high            218.62
low             123.02
close           171.51
volume   6949682394.00
dtype: float64
```

```
weather.query(
  'station == "GHCND:USW00094728"' # central park station
).pivot(columns='datatype', values='value')[['SNOW', 'PRCP']].sum() # finds the total snow an precipitation
```

```
datatype
SNOW   1007.00
PRCP   1665.30
dtype: float64
```

```
weather.query( # similar to code above
  'station == "GHCND:USW00094728"'
).pivot(columns='datatype', values='value')[['SNOW', 'PRCP']].agg('sum') # but using .agg() instead
```

```
datatype
SNOW   1007.00
PRCP   1665.30
dtype: float64
```

```
fb.agg({
  'open': 'mean',
  'high': ['min', 'max'], # passing a list, computing for min and max
  'low': ['min', 'max'], # returns a dataframe instead
  'close': 'mean' # but since there is only 1 operation is computed here, it would be filled with nan instead
})
```

	open	high	low	close
mean	171.45	NaN	NaN	171.51
min	NaN	129.74	123.02	NaN
max	NaN	218.62	214.27	NaN

```
# it is more ideal to aggregate on groups than on the entire dataframe
fb.groupby('trading_volume').mean() # recall groupby
```

trading_volume	open	high	low	close	volume
low	171.36	173.46	169.31	171.43	24547207.71
med	175.82	179.42	172.11	175.14	79072559.12
high	167.73	170.48	161.57	168.16	141924023.33

```
fb.groupby('trading_volume')['close'].agg(['min', 'max', 'mean'])
# selects the 'close' column and performs min, max and mean
```

trading_volume	min	max	mean
low	124.06	214.67	171.43
med	152.22	217.50	175.14
high	160.06	176.26	168.16

```
fb_agg = fb.groupby('trading_volume').agg({ # performing aggregate to multiple columns
  'open': 'mean',
  'high': ['min', 'max'], # and multiple operations
  'low': ['min', 'max'],
  'close': 'mean'
})
fb_agg # would result in a hierarchical index
```

	open	high		low		close
trading_volume	mean	min	max	min	max	mean
low	171.36	129.74	216.20	123.02	212.60	171.43
med	175.82	162.85	218.62	150.75	214.27	175.14
high	167.73	161.10	180.13	149.02	173.75	168.16

Next steps: View recommended plots

```
fb_agg.columns # shows the hierarchy
```

```
MultiIndex([( 'open', 'mean'),
            ( 'high',  'min'),
            ( 'high',  'max'),
            (  'low',  'min'),
            (  'low',  'max'),
            ('close', 'mean')],
           )
```

```
fb_agg.columns = ['_'.join(col_agg) for col_agg in fb_agg.columns] # joins the columns together to only have 1 index
fb_agg.head()
```

trading_volume	open_mean	high_min	high_max	low_min	low_max	close_mean
low	171.36	129.74	216.20	123.02	212.60	171.43
med	175.82	162.85	218.62	150.75	214.27	175.14
high	167.73	161.10	180.13	149.02	173.75	168.16

Next steps: View recommended plots

```
weather['2018-10'].query('datatype == "PRCP"').groupby( # gets the oct. 2018 precipitation data
  pd.Grouper(freq='D') # grouped by datetime
).mean().head()
```

```
<ipython-input-14-28b233bb899a>:1: FutureWarning: Indexing a DataFrame with a datetimelike index using a single string to slice the rows
  weather['2018-10'].query('datatype == "PRCP"').groupby( # gets the oct. 2018 precipitation data
<ipython-input-14-28b233bb899a>:3: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future
  ).mean().head()
```

date	value
2018-10-01	0.01
2018-10-02	2.23
2018-10-03	19.69
2018-10-04	0.32
2018-10-05	0.97

```
weather.query('datatype == "PRCP"').groupby(
  ['station_name', pd.Grouper(freq='Q')] # groups by quarter
).sum().unstack().sample(5, random_state=1)
```

```
<ipython-input-15-3de9e39f2b74>:3: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future v
  ).sum().unstack().sample(5, random_state=1)
```

	value			
date	2018-03-31	2018-06-30	2018-09-30	2018-12-31
station_name				
WANTAGH 1.1 NNE, NY US	279.90	216.80	472.50	277.20
STATEN ISLAND 1.4 SE, NY US	379.40	295.30	438.80	409.90
SYOSSET 2.0 SSW, NY US	323.50	263.30	355.50	459.90
STAMFORD 4.2 S, CT US	338.00	272.10	424.70	390.00
WAYNE TWP 0.8 SSW, NJ US	246.20	295.30	620.90	422.00

```
weather.groupby('station').filter( # filters stations
lambda x: 'NY' in x.name # with 'NY' in name
).query('datatype == "SNOW"').groupby('station_name').sum().squeeze() # gets snow data, groups by station name
```

```
<ipython-input-16-efd457d9b0e5>:3: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future v
  ).query('datatype == "SNOW"').groupby('station_name').sum().squeeze() # gets snow data, groups by station name
station_name
ALBERTSON 0.2 SSE, NY US         1087.00
AMITYVILLE 0.1 WSW, NY US         434.00
AMITYVILLE 0.6 NNE, NY US        1072.00
ARMONK 0.3 SE, NY US             1504.00
BROOKLYN 3.1 NW, NY US            305.00
CENTERPORT 0.9 SW, NY US          799.00
ELMSFORD 0.8 SSW, NY US           863.00
FLORAL PARK 0.4 W, NY US         1015.00
HICKSVILLE 1.3 ENE, NY US         716.00
JACKSON HEIGHTS 0.3 WSW, NY US    107.00
LOCUST VALLEY 0.3 E, NY US          0.00
LYNBROOK 0.3 NW, NY US            325.00
MASSAPEQUA 0.9 SSW, NY US          41.00
MIDDLE VILLAGE 0.5 SW, NY US     1249.00
NEW HYDE PARK 1.6 NE, NY US         0.00
NEW YORK 8.8 N, NY US               0.00
NORTH WANTAGH 0.4 WSW, NY US      471.00
PLAINEDGE 0.4 WSW, NY US          610.00
PLAINVIEW 0.4 ENE, NY US         1360.00
SADDLE ROCK 3.4 WSW, NY US        707.00
STATEN ISLAND 1.4 SE, NY US       936.00
STATEN ISLAND 4.5 SSE, NY US       89.00
SYOSSET 2.0 SSW, NY US           1039.00
VALLEY STREAM 0.6 SE, NY US       898.00
WANTAGH 0.3 ESE, NY US           1280.00
WANTAGH 1.1 NNE, NY US            940.00
WEST NYACK 1.3 WSW, NY US        1371.00
Name: value, dtype: float64
```

```
weather.query('datatype == "PRCP"').groupby( # gets precipitation data
  pd.Grouper(freq='D') # grouped by day
).mean().groupby(pd.Grouper(freq='M')).sum().value.nlargest() # finds the average and grouped by month
```

```
<ipython-input-17-c00b75e95382>:3: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future
  ).mean().groupby(pd.Grouper(freq='M')).sum().value.nlargest() # finds the average and grouped by month
date
2018-11-30   210.59
2018-09-30   193.09
2018-08-31   192.45
2018-07-31   160.98
2018-02-28   158.11
Name: value, dtype: float64
```

```
weather.query('datatype == "PRCP"').rename( # gets precipitaion data
  dict(value='prcp'), axis=1
).groupby(pd.Grouper(freq='D')).mean().groupby( # gets daily average
  pd.Grouper(freq='M') # groups by month
).transform(np.sum)['2018-01-28':'2018-02-03'] # uses .transfrom() to divide per month
```

```
<ipython-input-18-fe004da0d61c>:3: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future
  ).groupby(pd.Grouper(freq='D')).mean().groupby( # gets daily average
```

	prcp
date	
2018-01-28	69.31
2018-01-29	69.31
2018-01-30	69.31
2018-01-31	69.31
2018-02-01	158.11
2018-02-02	158.11
2018-02-03	158.11

```
# gets precipitation data
# renames to smallcase
# groups by day
# adds total and percent monthly precipitation
weather\
  .query('datatype == "PRCP"')\
  .rename(dict(value='prcp'), axis=1)\
  .groupby(pd.Grouper(freq='D')).mean()\
  .assign(
    total_prcp_in_month=lambda x: x.groupby(
      pd.Grouper(freq='M')
    ).transform(np.sum),
    pct_monthly_prcp=lambda x: x.prcp.div(
      x.total_prcp_in_month
    )
  ).nlargest(5, 'pct_monthly_prcp') # gets the 5 highest days with % monthly precipitation
```

```
<ipython-input-19-56be3f805285>:8: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future
  .groupby(pd.Grouper(freq='D')).mean()\
```

date	prcp	total_prcp_in_month	pct_monthly_prcp
2018-10-12	34.77	105.63	0.33
2018-01-13	21.66	69.31	0.31
2018-03-02	38.77	137.46	0.28
2018-04-16	39.34	140.57	0.28
2018-04-17	37.30	140.57	0.27

```
fb[['open', 'high', 'low', 'close']].transform(
  lambda x: (x - x.mean()).div(x.std()) # turns the data to z values
).head()
```

date	open	high	low	close
2018-01-02	0.32	0.41	0.41	0.50
2018-01-03	0.53	0.57	0.60	0.66
2018-01-04	0.68	0.65	0.74	0.64
2018-01-05	0.72	0.68	0.78	0.77
2018-01-08	0.80	0.79	0.85	0.84

```
fb.pivot_table(columns='trading_volume') # creates a pivot table
# a table that is centered around trading volume
```

trading_volume	low	med	high
close	171.43	175.14	168.16
high	173.46	179.42	170.48
low	169.31	172.11	161.57
open	171.36	175.82	167.73
volume	24547207.71	79072559.12	141924023.33

```
fb.pivot_table(index='trading_volume') # interchanges row and columns from code above
```

trading_volume	close	high	low	open	volume
low	171.43	173.46	169.31	171.36	24547207.71
med	175.14	179.42	172.11	175.82	79072559.12
high	168.16	170.48	161.57	167.73	141924023.33

```
weather.reset_index().pivot_table( # creates a pivot table
  index=['date', 'station', 'station_name'], # with 3 indices
  columns='datatype',
  values='value',
  aggfunc='median'
).reset_index().tail()
```

datatype	date	station	station_name	AWND	DAPR	MDPR	PGTM	PRCP	SNOW	SNWD	...	WSF5	WT01	WT02	WT03	WT04	WT05
28740	2018-12-31	GHCND:USW00054787	FARMINGDALE REPUBLIC AIRPORT, NY US	5.00	NaN	NaN	2052.00	28.70	NaN	NaN	...	15.70	NaN	NaN	NaN	NaN	NaN
28741	2018-12-31	GHCND:USW00094728	NY CITY CENTRAL PARK, NY US	NaN	NaN	NaN	NaN	25.90	0.00	0.00	...	NaN	1.00	NaN	NaN	NaN	NaN
28742	2018-12-31	GHCND:USW00094741	TETERBORO AIRPORT, NJ US	1.70	NaN	NaN	1954.00	29.20	NaN	NaN	...	8.90	NaN	NaN	NaN	NaN	NaN

```
pd.crosstab( # creates a frequency table with .crosstab() method
  index=fb.trading_volume,
  columns=fb.index.month,
  colnames=['month'] # named the frequency as month
)
```

month	1	2	3	4	5	6	7	8	9	10	11	12
trading_volume												
low	20	19	15	20	22	21	18	23	19	23	21	19
med	1	0	4	1	0	0	2	0	0	0	0	0

```
pd.crosstab(
  index=fb.trading_volume,
  columns=fb.index.month,
  colnames=['month'],
  normalize='columns' # normalize shows percentage instead (0.0 - 1.0)
)
```

month	1	2	3	4	5	6	7	8	9	10	11	12
trading_volume												
low	0.95	1.00	0.71	0.95	1.00	1.00	0.86	1.00	1.00	1.00	1.00	1.00
med	0.05	0.00	0.19	0.05	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00
high	0.00	0.00	0.10	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00

```
pd.crosstab(
  index=fb.trading_volume,
  columns=fb.index.month,
  colnames=['month'],
  values=fb.close, # performs on close column
  aggfunc=np.mean # the average
)
```

month	1	2	3	4	5	6	7	8	9	10	11	12
trading_volume												
low	185.24	180.27	177.07	163.29	182.93	195.27	201.92	177.49	164.38	154.19	141.64	137.16
med	179.37	NaN	164.76	174.16	NaN	NaN	194.28	NaN	NaN	NaN	NaN	NaN
high	NaN	NaN	164.11	NaN	NaN	NaN	176.26	NaN	NaN	NaN	NaN	NaN

```
snow_data = weather.query('datatype == "SNOW"') # gets snow column from weather
pd.crosstab(
  index=snow_data.station_name,
  columns=snow_data.index.month,
  colnames=['month'],
  values=snow_data.value,
  aggfunc=lambda x: (x > 0).sum(), # gets the total of positive values
  margins=True, # adds last column and row which are subtotals
  margins_name='total observations of snow'
)
```

station_name \ month	1	2	3	4	5	6	7	8	9	10	11	12	total observations of snow
ALBERTSON 0.2 SSE, NY US	3.00	1.00	3.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	9
AMITYVILLE 0.1 WSW, NY US	1.00	0.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3
AMITYVILLE 0.6 NNE, NY US	3.00	1.00	3.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8
ARMONK 0.3 SE, NY US	6.00	4.00	6.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	3.00	23
BLOOMINGDALE 0.7 SSE, NJ US	2.00	1.00	3.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	8
...	...	...	...	...	...	...	...	...	...	...	...	...	...
WESTFIELD 0.6 NE, NJ US	3.00	0.00	4.00	1.00	0.00	NaN	0.00	0.00	0.00	NaN	1.00	NaN	9
WOODBRIDGE TWP 1.1 ESE, NJ US	4.00	1.00	3.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	11
WOODBRIDGE TWP 1.1 NNE, NJ US	2.00	1.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	7
WOODBRIDGE TWP 3.0 NNW, NJ US	NaN	0.00	0.00	NaN	NaN	0.00	NaN	NaN	NaN	0.00	0.00	NaN	0