Get the average weekly market value of Bitcoin Cash.

```
data.loc[data.symbol == 'BCH', 'market'].resample('W').mean().head()
'Output':
date
2017-07-23    0.000000e+00
2017-07-30    0.000000e+00
2017-08-06    3.852961e+09
2017-08-13    4.982661e+09
2017-08-20    7.355117e+09
Freq: W-SUN, Name: market, dtype: float64
```

Convert to Datetime Datatype Using 'to_datetime'

Pandas uses the **to_datetime** method to convert strings to Pandas datetime datatype. The **to_datetime** method is smart enough to infer a **datetime** representation from a string of dates passed with different formats. The default output format of **to_datetime** is in the following order: **year, month, day, minute, second, millisecond, microsecond, nanosecond**.

The input to **to_datetime** is recognized as **month, day, year**. Although, it can easily be modified by setting the attributes **dayfirst** or **yearfirst** to **True**.

For example, if **dayfirst** is set to **True**, the input is recognized as **day, month, year**.

Let's see an example of this.

```
# create list of dates
my_dates = ['Friday, May 11, 2018', '11/5/2018', '11-5-2018', '5/11/2018',
            '2018.5.11']
pd.to_datetime(my_dates)
'Output':
DatetimeIndex(['2018-05-11', '2018-11-05', '2018-11-05', '2018-05-11',
               '2018-05-11'],
              dtype='datetime64[ns]', freq=None)
```

Let's set dayfirst to True. Observe that the first input in the string is treated as a day in the output.

```
# set dayfirst to True
pd.to_datetime('5-11-2018', dayfirst = True)
'Output':
Timestamp('2018-11-05 00:00:00')
```

The shift() Method

A typical step in a timeseries use case is to convert the timeseries dataset into a supervised learning framework for predicting the outcome for a given time instant. The **shift()** method is used to adjust a Pandas DataFrame column by shifting the observations forward or backward. If the observations are pulled backward (or lagged), **NaNs** are attached at the tail of the column. But if the values are pushed forward, the head of the column will contain **NaNs**. This step is important for adjusting the **target** variable of a dataset to predict outcomes *n*-days or steps or instances into the future. Let's see some examples.

Subset columns for the observations related to Bitcoin Cash.

```
# subset a few columns
data_subset_BCH = data.loc[data.symbol == 'BCH',
['open','high','low','close']]
data_subset_BCH.head()
'Output':
              open    high     low   close
date
2017-07-23  555.89  578.97  411.78  413.06
2017-07-24  412.58  578.89  409.21  440.70
2017-07-25  441.35  541.66  338.09  406.90
2017-07-26  407.08  486.16  321.79  365.82
2017-07-27  417.10  460.97  367.78  385.48
```

Now let's create a target variable that contains the closing rates 3 days into the future.

```
data_subset_BCH['close_4_ahead'] = data_subset_BCH['close'].shift(-4)
data_subset_BCH.head()
```