Learning_Pandas_Part_7_DateTimeOperations

June 20, 2021

0.0.1 Prepared by Abhishek Kumar

0.0.2 https://www.linkedin.com/in/abhishekkumar-0311/

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
# To get multiple outputs in the same cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

%matplotlib inline
```

[3]:
```
# Setup : DataFrame creation

salary = [['1','Abhishek Kumar','AIML', 'Machine Learning Engineer','M', 'Y',␣
 ↪'04051990', 1121000],
          ['2','Arjun Kumar','DM', 'Tech Lead','M', 'Y', '09031992', 109000],
          ['3','Vivek Raj','DM', 'Devops Engineer','M', 'N', np.NaN , 827000],
          ['4','Mika Singh','DM', 'Data Analyst','F', 'Y', '15101991',  np.NaN],
          ['5','Anusha Yenduri','AIML', 'Data Scientist','F', 'Y', '01011989', ␣
 ↪921000],
          ['6','Ritesh Srivastava','AIML', 'Data Engineer','M', 'Y', np.NaN,␣
 ↪785000]]

columns_name=['Emp_Id','Emp_Name','Department','Role','Gender', 'WFH Status',␣
 ↪'DOB', 'Salary']

emp_df = pd.DataFrame(salary,columns=columns_name)
emp_df
```

[3]:

	Emp_Id	Emp_Name	Department	Role	Gender	\
0	1	Abhishek Kumar	AIML	Machine Learning Engineer	M	
1	2	Arjun Kumar	DM	Tech Lead	M	
2	3	Vivek Raj	DM	Devops Engineer	M	
3	4	Mika Singh	DM	Data Analyst	F	

```
4      5     Anusha Yenduri       AIML             Data Scientist      F
5      6  Ritesh Srivastava       AIML              Data Engineer      M

  WFH Status       DOB     Salary
0          Y  04051990  1121000.0
1          Y  09031992   109000.0
2          N       NaN   827000.0
3          Y  15101991        NaN
4          Y  01011989   921000.0
5          Y       NaN   785000.0
```

[4]:
```
import numpy as np
import pandas as pd
sample = {
'col_a':['Houston,TX', 'Dallas,TX', 'Chicago,IL', 'Phoenix,AZ',      'San␣
↪Diego,CA'],
'col_b':['62K-70K', '62K-70K', '69K-76K', '62K-72K', '71K-78K' ],
'col_c':['A','B','A','a','c'],
'col_d':['  1x', ' 1y', '2x  ', '1x', '1y  ']
}
df_sample = pd.DataFrame(sample)
df_sample
```

[4]:
```
          col_a    col_b col_c col_d
0    Houston,TX  62K-70K     A    1x
1     Dallas,TX  62K-70K     B    1y
2    Chicago,IL  69K-76K     A  2x
3    Phoenix,AZ  62K-72K     a    1x
4  San Diego,CA  71K-78K     c  1y
```

0.0.3 WarmUp

[5]:
```
date = pd.to_datetime('12Apr2012')
date
type(date)
```

[5]:
```
Timestamp('2012-04-12 00:00:00')
```

[7]:
```
date = pd.to_timedelta(2)
date
```

[7]:
```
Timedelta('0 days 00:00:00.000000002')
```

[9]:
```
import datetime as dt
dtobj = dt.datetime.now()
dtobj
```

[9]:
```
datetime.datetime(2021, 5, 25, 22, 37, 9, 643423)
```

[10]:
```

```

[10]:
```
datetime.datetime(2021, 5, 25, 22, 37, 9, 657392)
```

[11]:
```
dtobj.year
dtobj.day
dtobj.month
dtobj.hour
dtobj.minute
dtobj.second
```

[11]:
```
25
```

1 0. Blogs

- https://towardsdatascience.com/working-with-datetime-in-pandas-dataframe-663f7af6c587
- https://www.analyticsvidhya.com/blog/2020/05/datetime-variables-python-pandas/
- https://towardsdatascience.com/mastering-dates-and-timestamps-in-pandas-and-python-in-general-5b8c6edcc50c

2 1. Reading / Converting to Timestamps

[17]:
```
df = pd.DataFrame({"month": [2, 3], "day": [4, 5], "hour": [2, 3],"year":␣
↪[2015, 2016]})
pd.to_datetime(df[["day", "month", "year"]])
```

[17]:
```
0   2015-02-04
1   2016-03-05
dtype: datetime64[ns]
```

[ ]:
```

```

3 2. Generating Date Ranges : `pd.date_range` , `pd.bdate_range`

- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.date_range.html#pandas.date_range
- `pandas.date_range(start=None, end=None, periods=None, freq=None, tz=None, normalize=False, name=None, closed=None, **kwargs)`[source]
- Parameters :
  - freq : possible values are **Y, M, D, W, Q, H, T for minutes, S**. Find more at https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#timeseries-offset-aliases

To make the creation of date sequences a convenient task, Pandas provides the date_range() method. It accepts a start date, an end date, and an optional frequency code:

If we need timestamps on a regular frequency, we can use the `date_range()` and `bdate_range()` functions to create a DatetimeIndex. The default frequency for `date_range` is a `calendar day` while the default for `bdate_range` is a `business day`:

`date_range` and `bdate_range` make it easy to generate a range of dates using various combinations of parameters like `start`, `end`, `periods`, and `freq`. The **start and end dates are strictly inclusive**, so dates outside of those specified will not be generated:

[18]:
```
pd.date_range(start='24/4/2020', end='24/5/2020', freq='D')
```

[18]:
```
DatetimeIndex(['2020-04-24', '2020-04-25', '2020-04-26', '2020-04-27',
               '2020-04-28', '2020-04-29', '2020-04-30', '2020-05-01',
               '2020-05-02', '2020-05-03', '2020-05-04', '2020-05-05',
               '2020-05-06', '2020-05-07', '2020-05-08', '2020-05-09',
               '2020-05-10', '2020-05-11', '2020-05-12', '2020-05-13',
               '2020-05-14', '2020-05-15', '2020-05-16', '2020-05-17',
               '2020-05-18', '2020-05-19', '2020-05-20', '2020-05-21',
               '2020-05-22', '2020-05-23', '2020-05-24'],
              dtype='datetime64[ns]', freq='D')
```

[19]:
```
pd.date_range(start='24/4/2020', end='24/5/2021', freq='M')
```

[19]:
```
DatetimeIndex(['2020-04-30', '2020-05-31', '2020-06-30', '2020-07-31',
               '2020-08-31', '2020-09-30', '2020-10-31', '2020-11-30',
               '2020-12-31', '2021-01-31', '2021-02-28', '2021-03-31',
               '2021-04-30'],
              dtype='datetime64[ns]', freq='M')
```

[20]:
```
pd.date_range(start='24/4/2020', end='24/5/2025', freq='Y')
```

[20]:
```
DatetimeIndex(['2020-12-31', '2021-12-31', '2022-12-31', '2023-12-31',
               '2024-12-31'],
              dtype='datetime64[ns]', freq='A-DEC')
```

[21]:
```
start_date = pd.to_datetime('today').date()
start_date
start_date = dt.datetime.now().date()
start_date
dates_end = pd.date_range(start=start_date, periods=10, freq='D')
dates_end
```

[21]:
```
datetime.date(2021, 5, 25)
```

[21]:
```
datetime.date(2021, 5, 25)
```

[21]:
```
DatetimeIndex(['2021-05-25', '2021-05-26', '2021-05-27', '2021-05-28',
               '2021-05-29', '2021-05-30', '2021-05-31', '2021-06-01',
               '2021-06-02', '2021-06-03'],
              dtype='datetime64[ns]', freq='D')
```

[ ]:
```
```

3.0.1 Creating a dataframe using date_range

[22]:
```
start_date = pd.to_datetime('today').date()
start_date
dates_end = pd.date_range(start=start_date, periods=10, freq='Y')
dates_end
```

[22]:
```
datetime.date(2021, 5, 25)
```

[22]:
```
DatetimeIndex(['2021-12-31', '2022-12-31', '2023-12-31', '2024-12-31',
               '2025-12-31', '2026-12-31', '2027-12-31', '2028-12-31',
               '2029-12-31', '2030-12-31'],
              dtype='datetime64[ns]', freq='A-DEC')
```

[23]:
```
pd.DataFrame(dates_end, columns = ['YearEndDates'])
```

[23]:

	YearEndDates
0	2021-12-31
1	2022-12-31
2	2023-12-31
3	2024-12-31
4	2025-12-31
5	2026-12-31
6	2027-12-31
7	2028-12-31
8	2029-12-31
9	2030-12-31

[24]:
```
start_date = pd.to_datetime('today').date()
start_date
dates_end = pd.date_range(end=start_date, periods=10, freq='Y')
dates_end
pd.DataFrame(dates_end, columns = ['YearEndDates'])
```

[24]:
```
datetime.date(2021, 5, 25)
```

[24]:
```
DatetimeIndex(['2011-12-31', '2012-12-31', '2013-12-31', '2014-12-31',
               '2015-12-31', '2016-12-31', '2017-12-31', '2018-12-31',
               '2019-12-31', '2020-12-31'],
              dtype='datetime64[ns]', freq='A-DEC')
```

[24]:

	YearEndDates
0	2011-12-31
1	2012-12-31
2	2013-12-31
3	2014-12-31
4	2015-12-31
5	2016-12-31
6	2017-12-31
7	2018-12-31
8	2019-12-31
9	2020-12-31

4 3. Indexing

[ ]:

5 4. Date/Time components

Property	Description
year	The year of the datetime
month	The month of the datetime
day	The days of the datetime
hour	The hour of the datetime
minute	The minutes of the datetime
second	The seconds of the datetime
microsecond	The microseconds of the datetime
nanosecond	The nanoseconds of the datetime
date	Returns datetime.date (does not contain timezone information)
time	Returns datetime.time (does not contain timezone information)
timetz	Returns datetime.time as local time with timezone information
dayofyear	The ordinal day of year
day_of_year	The ordinal day of year
weekofyear	The week ordinal of the year
week	The week ordinal of the year
dayofweek	The number of the day of the week with Monday=0, Sunday=6
day_of_week	The number of the day of the week with Monday=0, Sunday=6
weekday	The number of the day of the week with Monday=0, Sunday=6
quarter	Quarter of the date: Jan-Mar = 1, Apr-Jun = 2, etc.

Property	Description
days_in_month	The number of days in the month of the datetime
is_month_start	Logical indicating if first day of month (defined by frequency)
is_month_end	Logical indicating if last day of month (defined by frequency)
is_quarter_start	Logical indicating if first day of quarter (defined by frequency)
is_quarter_end	Logical indicating if last day of quarter (defined by frequency)
is_year_start	Logical indicating if first day of year (defined by frequency)
is_year_end	Logical indicating if last day of year (defined by frequency)
is_leap_year	Logical indicating if the date belongs to a leap year

5.1 Data Preparation

[25]:
```
df = pd.DataFrame({'date': ['2018-08-09 11:10:55','2019-03-02 13:15:21']})
df
df.dtypes
```

[25]:
```
                 date
0  2018-08-09 11:10:55
1  2019-03-02 13:15:21
```

[25]:
```
date    object
dtype: object
```

[26]:
```
# if column type is a string/object
# pd.DatetimeIndex(df['date']) returns Datetime type, which is chained with␣
 ↪strftime
df['yyyy_ww1'] = pd.DatetimeIndex(df['date']).strftime('%Y-%U')
df
df.dtypes
```

[26]:
```
                 date yyyy_ww1
0  2018-08-09 11:10:55  2018-31
1  2019-03-02 13:15:21  2019-08
```

[26]:
```
date        object
yyyy_ww1    object
dtype: object
```

5.1.1 .strftime()

- https://docs.python.org/3/library/datetime.html#strftime-and-strptime-behavior

[27]:
```
# if column type is a datetime
df['date'] = pd.to_datetime(df['date']) # Changing it to datatime object
df['yyyy_ww2'] = df['date'].dt.strftime('%Y-%U')
df
df.dtypes
```

[27]:
```
                 date yyyy_ww1 yyyy_ww2
0 2018-08-09 11:10:55  2018-31  2018-31
1 2019-03-02 13:15:21  2019-08  2019-08
```

[27]:
```
date        datetime64[ns]
yyyy_ww1            object
yyyy_ww2            object
dtype: object
```

[28]:
```
df.loc[len(df.index)] = [dt.datetime.now(),np.NaN,np.NaN]
df
```

[28]:
```
                        date yyyy_ww1 yyyy_ww2
0 2018-08-09 11:10:55.000000  2018-31  2018-31
1 2019-03-02 13:15:21.000000  2019-08  2019-08
2 2021-05-25 22:37:10.010439      NaN      NaN
```

- https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#time-date-components

[29]:
```
df['day'] = df['date'].dt.day
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year
df['hour'] = df['date'].dt.hour
df['minute'] = df['date'].dt.minute
df['second'] = df['date'].dt.second
df['microsecond'] = df['date'].dt.microsecond
df
df.dtypes
```

[29]:
```
                        date yyyy_ww1 yyyy_ww2  day  month  year  hour  \
0 2018-08-09 11:10:55.000000  2018-31  2018-31    9      8  2018    11
1 2019-03-02 13:15:21.000000  2019-08  2019-08    2      3  2019    13
2 2021-05-25 22:37:10.010439      NaN      NaN   25      5  2021    22

   minute  second  microsecond
0      10      55            0
1      15      21            0
```

```
2      37      10        10439
```

[29]:
```
date           datetime64[ns]
yyyy_ww1               object
yyyy_ww2               object
day                     int64
month                   int64
year                    int64
hour                    int64
minute                  int64
second                  int64
microsecond             int64
dtype: object
```

[30]:
```
df['datepart'] = df['date'].dt.date
df['timepart'] = df['date'].dt.time

df['weekday'] = df['date'].dt.weekday
df['dayofweek'] = df['date'].dt.dayofweek
df['dayofyear'] = df['date'].dt.dayofyear
df['weekofyear'] = df['date'].dt.isocalendar().week
df['quarter'] = df['date'].dt.quarter
df
df.dtypes
```

[30]:
```
                       date yyyy_ww1 yyyy_ww2  day  month  year  hour  \
0 2018-08-09 11:10:55.000000  2018-31  2018-31    9      8  2018    11
1 2019-03-02 13:15:21.000000  2019-08  2019-08    2      3  2019    13
2 2021-05-25 22:37:10.010439      NaN      NaN   25      5  2021    22

   minute  second  microsecond    datepart         timepart  weekday  \
0      10      55            0  2018-08-09         11:10:55        3
1      15      21            0  2019-03-02         13:15:21        5
2      37      10        10439  2021-05-25  22:37:10.010439        1

   dayofweek  dayofyear  weekofyear  quarter
0          3        221          32        3
1          5         61           9        1
2          1        145          21        2
```

[30]:
```
date           datetime64[ns]
yyyy_ww1               object
yyyy_ww2               object
day                     int64
month                   int64
year                    int64
hour                    int64
```

```
minute                  int64
second                  int64
microsecond             int64
datepart               object
timepart               object
weekday                 int64
dayofweek               int64
dayofyear               int64
weekofyear             UInt32
quarter                 int64
dtype: object
```

[31]:
```
df['is_month_start'] = df['date'].dt.is_month_start
df['is_month_end'] = df['date'].dt.is_month_end
df['is_year_start'] = df['date'].dt.is_year_start
df['is_year_end'] = df['date'].dt.is_year_end
df['is_leap_year'] = df['date'].dt.is_leap_year
df
df.dtypes
```

[31]:
```
                        date yyyy_ww1 yyyy_ww2  day  month  year  hour  \
0 2018-08-09 11:10:55.000000  2018-31  2018-31    9      8  2018    11
1 2019-03-02 13:15:21.000000  2019-08  2019-08    2      3  2019    13
2 2021-05-25 22:37:10.010439      NaN      NaN   25      5  2021    22

   minute  second  microsecond  … weekday dayofweek  dayofyear  weekofyear  \
0      10      55            0  …       3         3        221          32
1      15      21            0  …       5         5         61           9
2      37      10        10439  …       1         1        145          21

   quarter  is_month_start  is_month_end  is_year_start  is_year_end  \
0        3           False         False          False        False
1        1           False         False          False        False
2        2           False         False          False        False

   is_leap_year
0         False
1         False
2         False

[3 rows x 22 columns]
```

[31]:
```
date              datetime64[ns]
yyyy_ww1                  object
yyyy_ww2                  object
day                        int64
month                      int64
```

```
year                         int64
hour                         int64
minute                       int64
second                       int64
microsecond                  int64
datepart                    object
timepart                    object
weekday                      int64
dayofweek                    int64
dayofyear                    int64
weekofyear                  UInt32
quarter                      int64
is_month_start                bool
is_month_end                  bool
is_year_start                 bool
is_year_end                   bool
is_leap_year                  bool
dtype: object
```

- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.dt.day_name.html
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.dt.normalize.html
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.dt.round.html

[32]:
```
df['dayname'] = df['date'].dt.day_name()
df['monthname'] = df['date'].dt.month_name()
df['normalizeTime'] = df['date'].dt.normalize()
df['round'] = df['date'].dt.round(freq = "H")
df['floor'] = df['date'].dt.floor(freq = "H")
df['ceil'] = df['date'].dt.ceil(freq = "H")

df
df.dtypes
```

[32]:
```
                        date yyyy_ww1 yyyy_ww2  day  month  year  hour  \
0 2018-08-09 11:10:55.000000  2018-31  2018-31    9      8  2018    11
1 2019-03-02 13:15:21.000000  2019-08  2019-08    2      3  2019    13
2 2021-05-25 22:37:10.010439      NaN      NaN   25      5  2021    22

   minute  second  microsecond  … is_month_end is_year_start  is_year_end  \
0      10      55            0  …        False         False        False
1      15      21            0  …        False         False        False
2      37      10        10439  …        False         False        False

   is_leap_year   dayname  monthname  normalizeTime               round  \
0         False  Thursday     August     2018-08-09 2018-08-09 11:00:00
1         False  Saturday      March     2019-03-02 2019-03-02 13:00:00
2         False   Tuesday        May     2021-05-25 2021-05-25 23:00:00
```

```
                floor                ceil
0 2018-08-09 11:00:00 2018-08-09 12:00:00
1 2019-03-02 13:00:00 2019-03-02 14:00:00
2 2021-05-25 22:00:00 2021-05-25 23:00:00

[3 rows x 28 columns]
```

[32]:
```
date              datetime64[ns]
yyyy_ww1                  object
yyyy_ww2                  object
day                        int64
month                      int64
year                       int64
hour                       int64
minute                     int64
second                     int64
microsecond                int64
datepart                  object
timepart                  object
weekday                    int64
dayofweek                  int64
dayofyear                  int64
weekofyear                UInt32
quarter                    int64
is_month_start              bool
is_month_end                bool
is_year_start               bool
is_year_end                 bool
is_leap_year                bool
dayname                   object
monthname                 object
normalizeTime     datetime64[ns]
round             datetime64[ns]
floor             datetime64[ns]
ceil              datetime64[ns]
dtype: object
```

[33]:
```
df['mydate'] = pd.to_datetime(df[["day", "month", "year"]])
df
```

[33]:
```
                       date yyyy_ww1 yyyy_ww2  day  month  year  hour  \
0 2018-08-09 11:10:55.000000  2018-31  2018-31    9      8  2018    11
1 2019-03-02 13:15:21.000000  2019-08  2019-08    2      3  2019    13
2 2021-05-25 22:37:10.010439      NaN      NaN   25      5  2021    22

   minute  second  microsecond  … is_year_start is_year_end  is_leap_year  \
```

```
0      10      55            0  …         False       False         False
1      15      21            0  …         False       False         False
2      37      10        10439  …         False       False         False

    dayname  monthname  normalizeTime               round               floor  \
0  Thursday     August     2018-08-09 2018-08-09 11:00:00 2018-08-09 11:00:00
1  Saturday      March     2019-03-02 2019-03-02 13:00:00 2019-03-02 13:00:00
2   Tuesday        May     2021-05-25 2021-05-25 23:00:00 2021-05-25 22:00:00

                 ceil     mydate
0 2018-08-09 12:00:00 2018-08-09
1 2019-03-02 14:00:00 2019-03-02
2 2021-05-25 23:00:00 2021-05-25

[3 rows x 29 columns]
```

[ ]:

6 5. Date Offsets

- https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#dateoffset-objects

[34]:
```
ts = pd.Timestamp("2014-01-01 09:00")
ts
type(ts)
```

[34]:
```
Timestamp('2014-01-01 09:00:00')
```

[34]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

[35]:
```
ds = pd.to_datetime("2014-01-01")
ds
type(ds)
```

[35]:
```
Timestamp('2014-01-01 00:00:00')
```

[35]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

[36]:
```
ts.day_name()
ds.day_name()
```

[36]:
```
'Wednesday'
```

[36]:
```
'Wednesday'
```

6.0.1 Reset dataframe df

[41]:
```
df = pd.DataFrame({'date': ['2018-08-09 11:10:55','2019-03-02 13:15:21']})
df
df.dtypes
df.loc[len(df.index)] = [dt.datetime.now()]
df
df['date'] = pd.to_datetime(df['date'])
df
df.dtypes
```

[41]:
```
                  date
0  2018-08-09 11:10:55
1  2019-03-02 13:15:21
```

[41]:
```
date    object
dtype: object
```

[41]:
```
                           date
0          2018-08-09 11:10:55
1          2019-03-02 13:15:21
2  2021-05-25 22:43:17.024804
```

[41]:
```
                         date
0 2018-08-09 11:10:55.000000
1 2019-03-02 13:15:21.000000
2 2021-05-25 22:43:17.024804
```

[41]:
```
date    datetime64[ns]
dtype: object
```

- **https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.tseries.offsets.DateOffse**

[45]:
```
df['Off_Years'] =  df['date'] + pd.DateOffset(years=2)
df['Off_Months'] =  df['date'] + pd.DateOffset(months=2)
df['Off_Days'] =  df['date'] + pd.DateOffset(days=2)
df['Off_Weeks'] =  df['date'] + pd.DateOffset(weeks=2)
df['Off_Hours'] =  df['date'] + pd.DateOffset(hours=2)
df['Off_Mins'] =  df['date'] + pd.DateOffset(minutes=2)
df['Off_Secs'] =  df['date'] + pd.DateOffset(seconds=2)
df['Off_MilliSecs'] =  df['date'] + pd.DateOffset(milliseconds=2)
df
df.dtypes
```

[45]:
```
                         date                  Off_Years  \
0 2018-08-09 11:10:55.000000 2020-08-09 11:10:55.000000
1 2019-03-02 13:15:21.000000 2021-03-02 13:15:21.000000
2 2021-05-25 22:43:17.024804 2023-05-25 22:43:17.024804
```

```
                  Off_Months                  Off_Days  \
0 2018-10-09 11:10:55.000000 2018-08-11 11:10:55.000000
1 2019-05-02 13:15:21.000000 2019-03-04 13:15:21.000000
2 2021-07-25 22:43:17.024804 2021-05-27 22:43:17.024804

                   Off_Weeks                  Off_Hours  \
0 2018-08-23 11:10:55.000000 2018-08-09 13:10:55.000000
1 2019-03-16 13:15:21.000000 2019-03-02 15:15:21.000000
2 2021-06-08 22:43:17.024804 2021-05-26 00:43:17.024804

                    Off_Mins                  Off_Secs  \
0 2018-08-09 11:12:55.000000 2018-08-09 11:10:57.000000
1 2019-03-02 13:17:21.000000 2019-03-02 13:15:23.000000
2 2021-05-25 22:45:17.024804 2021-05-25 22:43:19.024804

               Off_MilliSecs
0 2018-08-09 11:10:55.002000
1 2019-03-02 13:15:21.002000
2 2021-05-25 22:43:17.026804
```

[45]:
```
date             datetime64[ns]
Off_Years        datetime64[ns]
Off_Months       datetime64[ns]
Off_Days         datetime64[ns]
Off_Weeks        datetime64[ns]
Off_Hours        datetime64[ns]
Off_Mins         datetime64[ns]
Off_Secs         datetime64[ns]
Off_MilliSecs    datetime64[ns]
dtype: object
```

6.0.2 Reset dataframe df

[49]:
```
df = pd.DataFrame({'date': ['2018-08-09 11:10:55','2019-03-02 13:15:21']})
df
df.dtypes
df.loc[len(df.index)] = [dt.datetime.now()]
df
df['date'] = pd.to_datetime(df['date'])
df
df.dtypes
```

[49]:
```
                  date
0  2018-08-09 11:10:55
1  2019-03-02 13:15:21
```

[49]:
```
date    object
dtype: object
```

[49]:
```
                        date
0         2018-08-09 11:10:55
1         2019-03-02 13:15:21
2  2021-05-25 23:05:04.163623
```

[49]:
```
                        date
0 2018-08-09 11:10:55.000000
1 2019-03-02 13:15:21.000000
2 2021-05-25 23:05:04.163623
```

[49]:
```
date    datetime64[ns]
dtype: object
```

6.0.3 The below code increment/decrement the date values based on +ve/-ve signs.

The Offset Unit is provided as method

- **https://pandas.pydata.org/pandas-docs/stable/reference/offset_frequency.html**

[50]:
```
df['Off_Years_End'] =  df['date'] + -2*pd.offsets.YearEnd()
df['Off_Years_Begin'] =  df['date'] + 2*pd.offsets.YearBegin()
df['Off_Months_End'] =  df['date'] + 2*pd.offsets.MonthEnd()
df['Off_Months_Begin'] =  df['date'] + 2*pd.offsets.MonthBegin()
df['Off_Quarter_End'] =  df['date'] + 2*pd.offsets.QuarterEnd()
df['Off_Quarter_Begin'] =  df['date'] + 2*pd.offsets.QuarterBegin()
df['Off_Weeks'] =  df['date'] + 2*pd.offsets.Week()
df['Off_Days'] =  df['date'] + 2*pd.offsets.Day()
df['Off_Bdays'] =  df['date'] + 2*pd.offsets.BDay()
df['Off_Hours'] =  df['date'] + 2*pd.offsets.Hour()
df['Off_Mins'] =  df['date'] + 2*pd.offsets.Minute()
df['Off_Secs'] =  df['date'] + 2*pd.offsets.Second()


df
df.dtypes
```

[50]:
```
                        date              Off_Years_End  \
0 2018-08-09 11:10:55.000000 2016-12-31 11:10:55.000000
1 2019-03-02 13:15:21.000000 2017-12-31 13:15:21.000000
2 2021-05-25 23:05:04.163623 2019-12-31 23:05:04.163623

             Off_Years_Begin             Off_Months_End  \
0 2020-01-01 11:10:55.000000 2018-09-30 11:10:55.000000
1 2021-01-01 13:15:21.000000 2019-04-30 13:15:21.000000
2 2023-01-01 23:05:04.163623 2021-06-30 23:05:04.163623
```

```
            Off_Months_Begin            Off_Quarter_End  \
0 2018-10-01 11:10:55.000000 2018-12-31 11:10:55.000000
1 2019-05-01 13:15:21.000000 2019-06-30 13:15:21.000000
2 2021-07-01 23:05:04.163623 2021-09-30 23:05:04.163623

           Off_Quarter_Begin                 Off_Weeks  \
0 2018-12-01 11:10:55.000000 2018-08-23 11:10:55.000000
1 2019-09-01 13:15:21.000000 2019-03-16 13:15:21.000000
2 2021-09-01 23:05:04.163623 2021-06-08 23:05:04.163623

                    Off_Days                 Off_Bdays  \
0 2018-08-11 11:10:55.000000 2018-08-13 11:10:55.000000
1 2019-03-04 13:15:21.000000 2019-03-05 13:15:21.000000
2 2021-05-27 23:05:04.163623 2021-05-27 23:05:04.163623

                   Off_Hours                  Off_Mins  \
0 2018-08-09 13:10:55.000000 2018-08-09 11:12:55.000000
1 2019-03-02 15:15:21.000000 2019-03-02 13:17:21.000000
2 2021-05-26 01:05:04.163623 2021-05-25 23:07:04.163623

                    Off_Secs
0 2018-08-09 11:10:57.000000
1 2019-03-02 13:15:23.000000
2 2021-05-25 23:05:06.163623
```

[50]:
```
date                datetime64[ns]
Off_Years_End       datetime64[ns]
Off_Years_Begin     datetime64[ns]
Off_Months_End      datetime64[ns]
Off_Months_Begin    datetime64[ns]
Off_Quarter_End     datetime64[ns]
Off_Quarter_Begin   datetime64[ns]
Off_Weeks           datetime64[ns]
Off_Days            datetime64[ns]
Off_Bdays           datetime64[ns]
Off_Hours           datetime64[ns]
Off_Mins            datetime64[ns]
Off_Secs            datetime64[ns]
dtype: object
```

[ ]: