In [3]:

```
#Module import
import pandas as pd
import numpy as np
import datetime as dt
from datetime import timedelta
import time
# from pandas.tseries.offsets import Hour,Minute,Day,MonthEnd,Year,weekend
from pandas.tseries import offsets
from ast import literal_eval
```

Data

In [4]:

```
ts =pd.Series(np.random.randn(20),
              index=pd.date_range('1/15/2000',periods=20,freq='4d'))
print(ts)
type(ts)
```

```
2000-01-15   -0.760744
2000-01-19    0.572903
2000-01-23   -0.158549
2000-01-27   -0.656381
2000-01-31   -0.652676
2000-02-04    0.209360
2000-02-08    0.187652
2000-02-12   -0.945363
2000-02-16   -2.191602
2000-02-20    0.062052
2000-02-24   -1.602248
2000-02-28   -0.961349
2000-03-03   -1.863905
2000-03-07    0.447988
2000-03-11    0.812511
2000-03-15    0.487992
2000-03-19   -0.666159
2000-03-23   -1.230986
2000-03-27    0.684753
2000-03-31   -0.097564
Freq: 4D, dtype: float64
```

```
pandas.core.series.Series
```

offset.rollforward 기준으로 그룹화를 진행 후 그룹별 평균값을 출력하세요

In [5]:

```
# offset.rollforward
# offset을 기준으로 한다...?

offset_do = offsets.DateOffset()
grouped_do = ts.groupby([offset_do.rollforward]).mean()
print('grouped_do','\n',grouped_do.head())
print()

offset_ME = offsets.MonthEnd()
grouped_ME = ts.groupby([offset_ME.rollforward]).mean()
print('grouped_ME','\n',grouped_ME)
print()

offset_D = offsets.Day()
grouped_D = ts.groupby([offset_D.rollforward]).mean()
print('grouped_D','\n',grouped_D.head())
print()
offset_Y = offsets.YearEnd()
grouped_Y = ts.groupby([offset_Y.rollforward]).mean()
print('grouped_Y','\n',grouped_Y.head())
print() #연도가 2000년도 하나라 값이 하나뿐

offset_W = offsets.Week()
grouped_W = ts.groupby([offset_W.rollforward]).mean()
print('grouped_W','\n',grouped_W.head())
print()

offset_Wm = offsets.WeekOfMonth()
grouped_Wm = ts.groupby([offset_Wm.rollforward]).mean()
print('grouped_Wm','\n',grouped_Wm.head())
print()


#등등등 엄청 많다
```

```
2000-01-23   -0.158549
2000-01-27   -0.656381
2000-01-31   -0.652676
dtype: float64

grouped_Y
 2000-12-31   -0.416116
dtype: float64

grouped_W
 2000-01-15   -0.760744
2000-01-19    0.572903
2000-01-23   -0.158549
2000-01-27   -0.656381
2000-01-31   -0.652676
dtype: float64

grouped_Wm
```

```
 2000-02-07   -0.241014
2000-03-06   -1.044966
2000-04-03    0.062648
dtype: float64
```

상기 문제를 resample 을 사용하여 수행하세요

In [6]:

```
r_s = ts.resample('D').mean()
r_s
```

```
2000-01-15   -0.760744
2000-01-16         NaN
2000-01-17         NaN
2000-01-18         NaN
2000-01-19    0.572903
                ...
2000-03-27    0.684753
2000-03-28         NaN
2000-03-29         NaN
2000-03-30         NaN
2000-03-31   -0.097564
Freq: D, Length: 77, dtype: float64
```

위와 동일!

날짜 데이터를 인덱스로 하는 5가지 이상의 컬럼을 포함하고 있는 데이터 셋을 생성하세요.

단 각 컬럼은 인덱스 별 의미있는 값을 갖도록 만든 후 다음을 수행하세요
1. 특정 연별, 월별, 일별 특정 컬럼 값 집계
2. 인덱스를 실수형으로 변환 (timestamp) 후 다시 datetime 으로 변환
3. 인덱스의 포멧으로 변환

"날짜 데이터를 인덱스로 하는 5가지 이상의 컬럼을 포함하고 있는 dataset을 생성"

In [7]:

```
raw_data = pd.read_csv('../Data/income.csv')
df = raw_data.copy()
df.columns
df.head()
```

	일자	관광수지(백만 $)	관광수입(백만 $)	수입1인당(달러)	관광지출	지출1인당(달러)
0	2020-01-01	-1,008.00	1,459.10	1,146.50	2,467.10	981.7
1	2020-02-01	-272.4	1,054.20	1,538.50	1,326.60	1,267.30
2	2020-03-01	-148.3	889.3	10,650.70	1,037.60	7,237.40
3	2020-04-01	-157.7	631.6	21,472.00	789.3	25,116.90
4	2020-05-01	-15.6	663.9	21,551.00	679.5	17,975.20

In [8]:

```
df = df.rename(columns={"일자":"Date","관광수지(백만$)":"total","관광수입(백만$
df
```

	Date	total	t_income	ic_p_person	t_outcome	oc_p_person
0	2020-01-01	-1,008.00	1,459.10	1,146.50	2,467.10	981.7
1	2020-02-01	-272.4	1,054.20	1,538.50	1,326.60	1,267.30
2	2020-03-01	-148.3	889.3	10,650.70	1,037.60	7,237.40
3	2020-04-01	-157.7	631.6	21,472.00	789.3	25,116.90
4	2020-05-01	-15.6	663.9	21,551.00	679.5	17,975.20
5	2020-06-01	-227.2	681	18,436.30	908.2	18,782.70
6	2020-07-01	-133.5	815.1	13,359.70	948.6	14,386.70
7	2020-08-01	-119.6	884.4	12,855.20	1,004.00	11,295.10
8	2020-09-01	-139	862.2	13,256.50	1,001.20	13,036.80
9	2020-10-01	-223.3	851.8	13,831.30	1,075.10	14,938.20
10	2020-11-01	-281.1	856.9	13,873.80	1,138.00	16,099.40
11	2020-12-01	-288	786.7	12,618.70	1,074.70	13,272.30
12	2021-01-01	-268.5	874.6	14,976.80	1,143.10	13,269.80
13	2021-02-01	-193.4	692.2	10,554.70	885.6	12,982.90
14	2021-03-01	-203.6	850.6	11,401.50	1,054.20	14,246.10
15	2021-04-01	-411.3	812.5	11,588.60	1,223.80	17,163.60
16	2021-05-01	-514	845.4	11,353.30	1,359.40	18,025.40

In [9]:

```
df
```

	Date	total	t_income	ic_p_person	t_outcome	oc_p_person
0	2020-01-01	-1,008.00	1,459.10	1,146.50	2,467.10	981.7
1	2020-02-01	-272.4	1,054.20	1,538.50	1,326.60	1,267.30
2	2020-03-01	-148.3	889.3	10,650.70	1,037.60	7,237.40
3	2020-04-01	-157.7	631.6	21,472.00	789.3	25,116.90
4	2020-05-01	-15.6	663.9	21,551.00	679.5	17,975.20
5	2020-06-01	-227.2	681	18,436.30	908.2	18,782.70
6	2020-07-01	-133.5	815.1	13,359.70	948.6	14,386.70
7	2020-08-01	-119.6	884.4	12,855.20	1,004.00	11,295.10
8	2020-09-01	-139	862.2	13,256.50	1,001.20	13,036.80
9	2020-10-01	-223.3	851.8	13,831.30	1,075.10	14,938.20
10	2020-11-01	-281.1	856.9	13,873.80	1,138.00	16,099.40
11	2020-12-01	-288	786.7	12,618.70	1,074.70	13,272.30
12	2021-01-01	-268.5	874.6	14,976.80	1,143.10	13,269.80
13	2021-02-01	-193.4	692.2	10,554.70	885.6	12,982.90
14	2021-03-01	-203.6	850.6	11,401.50	1,054.20	14,246.10
15	2021-04-01	-411.3	812.5	11,588.60	1,223.80	17,163.60
16	2021-05-01	-514	845.4	11,353.30	1,359.40	18,025.40

In [10]:

```
# year, month, day 로 나눠주기 위해선, datetime형식으로 바꾸어 주어야 됨.
df['dt_Date'] = pd.to_datetime(df['Date'])
df[['Date','dt_Date']].info() #datetime으로 변환 완료
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17 entries, 0 to 16
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype         
---  ------   --------------  -----         
 0   Date     17 non-null     object        
 1   dt_Date  17 non-null     datetime64[ns]
dtypes: datetime64[ns](1), object(1)
memory usage: 400.0+ bytes
```

In [11]:

```
df['Year'] = df['dt_Date'].dt.year
df['Month'] = df['dt_Date'].dt.month
df['Day'] = df['dt_Date'].dt.day
df[['Date','dt_Date','Year','Month','Day']]
```

	Date	dt_Date	Year	Month	Day
0	2020-01-01	2020-01-01	2020	1	1
1	2020-02-01	2020-02-01	2020	2	1
2	2020-03-01	2020-03-01	2020	3	1
3	2020-04-01	2020-04-01	2020	4	1
4	2020-05-01	2020-05-01	2020	5	1
5	2020-06-01	2020-06-01	2020	6	1
6	2020-07-01	2020-07-01	2020	7	1
7	2020-08-01	2020-08-01	2020	8	1
8	2020-09-01	2020-09-01	2020	9	1
9	2020-10-01	2020-10-01	2020	10	1
10	2020-11-01	2020-11-01	2020	11	1
11	2020-12-01	2020-12-01	2020	12	1
12	2021-01-01	2021-01-01	2021	1	1
13	2021-02-01	2021-02-01	2021	2	1
14	2021-03-01	2021-03-01	2021	3	1
15	2021-04-01	2021-04-01	2021	4	1
16	2021-05-01	2021-05-01	2021	5	1

In [12]:

```
df = df.drop('Date',axis=1)
```

In [13]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17 entries, 0 to 16
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype         
---  ------       --------------  -----         
 0   total        17 non-null     object        
 1   t_income     17 non-null     object        
 2   ic_p_person  17 non-null     object        
 3   t_outcome    17 non-null     object        
 4   oc_p_person  17 non-null     object        
 5   dt_Date      17 non-null     datetime64[ns]
 6   Year         17 non-null     int64         
 7   Month        17 non-null     int64         
 8   Day          17 non-null     int64         
dtypes: datetime64[ns](1), int64(3), object(5)
memory usage: 1.3+ KB
```

In [14]:

```
df = df.set_index('dt_Date')
df
```

dt_Date	total	t_income	ic_p_person	t_outcome	oc_p_person	Year	Month	D
2020-01-01	-1,008.00	1,459.10	1,146.50	2,467.10	981.7	2020	1	1
2020-02-01	-272.4	1,054.20	1,538.50	1,326.60	1,267.30	2020	2	1
2020-03-01	-148.3	889.3	10,650.70	1,037.60	7,237.40	2020	3	1
2020-04-01	-157.7	631.6	21,472.00	789.3	25,116.90	2020	4	1
2020-05-01	-15.6	663.9	21,551.00	679.5	17,975.20	2020	5	1
2020-06-01	-227.2	681	18,436.30	908.2	18,782.70	2020	6	1
2020-07-01	-133.5	815.1	13,359.70	948.6	14,386.70	2020	7	1
2020-08-01	-119.6	884.4	12,855.20	1,004.00	11,295.10	2020	8	1
2020-09-01	-139	862.2	13,256.50	1,001.20	13,036.80	2020	9	1
2020-10-01	-223.3	851.8	13,831.30	1,075.10	14,938.20	2020	10	1
2020-11-01	-281.1	856.9	13,873.80	1,138.00	16,099.40	2020	11	1
2020-12-01	-288	786.7	12,618.70	1,074.70	13,272.30	2020	12	1
2021-01-01	-268.5	874.6	14,976.80	1,143.10	13,269.80	2021	1	1
2021-02-01	-193.4	692.2	10,554.70	885.6	12,982.90	2021	2	1
2021-03-01	-203.6	850.6	11,401.50	1,054.20	14,246.10	2021	3	1
2021-04-01	-411.3	812.5	11,588.60	1,223.80	17,163.60	2021	4	1
2021-05-01	-514	845.4	11,353.30	1,359.40	18,025.40	2021	5	1

1. 특정 연별, 월별, 일별 특정 컬럼 값 집계

In [15]:

```
# 반장님 감사합니다.
# 추가과제가 생겼습니다.
# 제가 한번 해결해보겠습니다.
# 꼭 올리겠습니다!

#20210721 100300
# 해결완료.
# dollar 의 경우에는 쉽게 안바뀌네요.
# 정규표현식 사용으로 바꾸어 줍니다.
import re
def dollar_to_float(x):
    new=re.sub(',',"",x)
    new=float(new)
    return new
df['total'] =df['total'].apply(lambda x : dollar_to_float(x))
df['total']
```

```
dt_Date
2020-01-01   -1008.0
2020-02-01    -272.4
2020-03-01    -148.3
2020-04-01    -157.7
2020-05-01     -15.6
2020-06-01    -227.2
2020-07-01    -133.5
2020-08-01    -119.6
2020-09-01    -139.0
2020-10-01    -223.3
2020-11-01    -281.1
2020-12-01    -288.0
2021-01-01    -268.5
2021-02-01    -193.4
2021-03-01    -203.6
2021-04-01    -411.3
2021-05-01    -514.0
Name: total, dtype: float64
```

In [17]:

```
print('연간 관광 수지',df.total.resample('1Y').sum(),"", sep='\n') #데이터부족
print('월간 관광 수지',df.total.resample('1M').sum())
print('일별 관광 수지',df.total.resample('1D').sum().head()) #데이터 부족
```

```
연간 관광 수지
dt_Date
2020-12-31   -3013.7
2021-12-31   -1590.8
Freq: A-DEC, Name: total, dtype: float64

월간 관광 수지 dt_Date
2020-01-31   -1008.0
2020-02-29    -272.4
2020-03-31    -148.3
2020-04-30    -157.7
2020-05-31     -15.6
2020-06-30    -227.2
2020-07-31    -133.5
2020-08-31    -119.6
2020-09-30    -139.0
2020-10-31    -223.3
2020-11-30    -281.1
2020-12-31    -288.0
2021-01-31    -268.5
2021-02-28    -193.4
2021-03-31    -203.6
2021-04-30    -411.3
2021-05-31    -514.0
Freq: M, Name: total, dtype: float64
일별 관광 수지 dt_Date
2020-01-01   -1008.0
2020-01-02       0.0
2020-01-03       0.0
2020-01-04       0.0
2020-01-05       0.0
Freq: D, Name: total, dtype: float64
```

2. 인덱스를 실수형으로 변환 (timestamp) 후 다시 datetime 으로 변환

In [15]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 17 entries, 2020-01-01 to 2021-05-01
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   total        17 non-null     object
 1   t_income     17 non-null     object
 2   ic_p_person  17 non-null     object
 3   t_outcome    17 non-null     object
 4   oc_p_person  17 non-null     object
 5   Year         17 non-null     int64 
 6   Month        17 non-null     int64 
 7   Day          17 non-null     int64 
 8   total_n      17 non-null     object
dtypes: int64(3), object(6)
memory usage: 1.3+ KB
```

In [16]:

```
print(type(df.index)) #datetimeIndex 형태
#인덱스에서 제거

df = df.reset_index()
print(df['dt_Date'],type(df['dt_Date'])) #datetime 형태로 변환


# 여기서 잠깐!
# mktime 함수는 time.struct_time (timetuple)로 바꾸던지 timestamp(epoch time ;
# 하지만 mktime의 경우 Series / DataFrame 에 적용이 되지 않는다.
# 실수형으로 바꾸고 싶다면 Array 내의 요소들 하나씩을 바꿔줘야 한다.
# 이는 곧 APPLY 함수 + lambda 세트를 활용하면!
# 가!
# 능!


df['ts_Date(f)'] = df['dt_Date'].apply(lambda x : time.mktime(x.timetuple())) #
print()
# timestamp로 변환 - mktime / timetuple
print(df['ts_Date(f)'],type(df['ts_Date(f)']))
```

```
9    2020-10-01
10   2020-11-01
11   2020-12-01
12   2021-01-01
13   2021-02-01
14   2021-03-01
15   2021-04-01
16   2021-05-01
Name: dt_Date, dtype: datetime64[ns] <class 'pandas.core.series.Series'>

0     1.577804e+09
1     1.580483e+09
2     1.582988e+09
3     1.585667e+09
4     1.588259e+09
5     1.590937e+09
6     1.593529e+09
7     1.596208e+09
8     1.598886e+09
9     1.601478e+09
10    1.604156e+09
11    1.606748e+09
```

In [17]:

```
df[['dt_Date','ts_Date(f)']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17 entries, 0 to 16
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   dt_Date     17 non-null     datetime64[ns]
 1   ts_Date(f)  17 non-null     float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 400.0 bytes
```

In [18]:

```
#민찬님의 과제를 몰래 훔쳐보고 참고한 결과
#numeric 함수가 존재...!
#스칼라값 / 리스트 / 튜플 / 배열 / 시리즈에 적용 가능!
df['numer(i)'] = pd.to_numeric(df['dt_Date']) #정수형으로 변화?
```

In [19]:

```
# to_datetime(바꾸고 싶은 요소(array/series) datetime 형태로.
```

In [20]:

```
df[['dt_Date','ts_Date(f)','numer(i)']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17 entries, 0 to 16
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   dt_Date     17 non-null     datetime64[ns]
 1   ts_Date(f)  17 non-null     float64       
 2   numer(i)    17 non-null     int64         
dtypes: datetime64[ns](1), float64(1), int64(1)
memory usage: 536.0 bytes
```

In [21]:

```
df.set_index(['dt_Date'],inplace=True)
```

In [22]:

```
df.head()
```

dt_Date	total	t_income	ic_p_person	t_outcome	oc_p_person	Year	Month	D
2020-01-01	-1,008.00	1,459.10	1,146.50	2,467.10	981.7	2020	1	1
2020-02-01	-272.4	1,054.20	1,538.50	1,326.60	1,267.30	2020	2	1
2020-03-01	-148.3	889.3	10,650.70	1,037.60	7,237.40	2020	3	1
2020-04-01	-157.7	631.6	21,472.00	789.3	25,116.90	2020	4	1
2020-05-01	-15.6	663.9	21,551.00	679.5	17,975.20	2020	5	1

3. 인덱스를 포멧으로 변환

In [23]:

```
df.index = df.index.strftime('%y-%m-%d')
df
#% Y  0000 (2021)
#% m  05  07
```

dt_Date	total	t_income	ic_p_person	t_outcome	oc_p_person	Year	Month	D
20-01-01	-1,008.00	1,459.10	1,146.50	2,467.10	981.7	2020	1	1
20-02-01	-272.4	1,054.20	1,538.50	1,326.60	1,267.30	2020	2	1
20-03-01	-148.3	889.3	10,650.70	1,037.60	7,237.40	2020	3	1
20-04-01	-157.7	631.6	21,472.00	789.3	25,116.90	2020	4	1
20-05-01	-15.6	663.9	21,551.00	679.5	17,975.20	2020	5	1
20-06-01	-227.2	681	18,436.30	908.2	18,782.70	2020	6	1
20-07-01	-133.5	815.1	13,359.70	948.6	14,386.70	2020	7	1
20-08-01	-119.6	884.4	12,855.20	1,004.00	11,295.10	2020	8	1
20-09-01	-139	862.2	13,256.50	1,001.20	13,036.80	2020	9	1
20-10-01	-223.3	851.8	13,831.30	1,075.10	14,938.20	2020	10	1
20-11-01	-281.1	856.9	13,873.80	1,138.00	16,099.40	2020	11	1
20-12-01	-288	786.7	12,618.70	1,074.70	13,272.30	2020	12	1
21-01-01	-268.5	874.6	14,976.80	1,143.10	13,269.80	2021	1	1
21-02-01	-193.4	692.2	10,554.70	885.6	12,982.90	2021	2	1
21-03-01	-203.6	850.6	11,401.50	1,054.20	14,246.10	2021	3	1
21-04-01	-411.3	812.5	11,588.60	1,223.80	17,163.60	2021	4	1
21-05-01	-514	845.4	11,353.30	1,359.40	18,025.40	2021	5	1

In [ ]: