1 第十章：分割应用组合

数据小鱼Rexa
CSDN：https://blog.csdn.net/qq_38395376?spm=1011.2124.3001.5343
Blibli：https://space.bilibili.com/283181288
Github：https://github.com/Rexa-Yu

1.1 基本的单变量分组聚合

In [379]:

```
# 加载数据
import pandas as pd
df=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\data/gapminder.tsv",s
df.head()
```

Out[379]:

	country	continent	year	lifeExp	pop	gdpPercap
0	Afghanistan	Asia	1952	28.801	8425333	779.445314
1	Afghanistan	Asia	1957	30.332	9240934	820.853030
2	Afghanistan	Asia	1962	31.997	10267083	853.100710
3	Afghanistan	Asia	1967	34.020	11537966	836.197138
4	Afghanistan	Asia	1972	36.088	13079460	739.981106

In [380]:

```
# 获取年份唯一的列表
years=df.year.unique()
years
```

Out[380]:

```
array([1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, 200
2,
       2007], dtype=int64)
```

In [381]:

```
# 求1952年的平均寿命
y1952_mean=df.loc[df.year==1952,:].lifeExp.mean()
y1952_mean
```

Out[381]:

```
49.05761971830987
```

1.2 聚合函数agg和aggregate

In [382]:

```
# 简单的agg使用
# agg需要配合numpy的函数使用
import numpy as np
cont_le_agg=df.groupby("continent").lifeExp.agg(np.mean)
cont_le_agg
```

Out[382]:

```
continent
Africa      48.865330
Americas    64.658737
Asia        60.064903
Europe      71.903686
Oceania     74.326208
Name: lifeExp, dtype: float64
```

In [383]:

```
import numpy as np
cont_le_agg=df.groupby("continent").lifeExp.aggregate(np.mean)
cont_le_agg
```

Out[383]:

```
continent
Africa      48.865330
Americas    64.658737
Asia        60.064903
Europe      71.903686
Oceania     74.326208
Name: lifeExp, dtype: float64
```

In [384]:

```
# 定义自己的函数，可以通过aggregate进行调用
def my_mean(values):
    n=len(values)
    sum=0
    for i in values:
        sum+=i
    return (sum/n)
agg_my_mean=df.groupby("year").lifeExp.aggregate(my_mean)
agg_my_mean
```

Out[384]:

```
year
1952    49.057620
1957    51.507401
1962    53.609249
1967    55.678290
1972    57.647386
1977    59.570157
1982    61.533197
1987    63.212613
1992    64.160338
1997    65.014676
2002    65.694923
2007    67.007423
Name: lifeExp, dtype: float64
```

In [385]:

```
# 同时传入多个函数
# 计算lifeexp的非零个数，平均值、和标准差
gdf=df.groupby("year").lifeExp.agg([np.count_nonzero,np.mean,np.std])
gdf
```

Out[385]:

	count_nonzero	mean	std
year			
1952	142.0	49.057620	12.225956
1957	142.0	51.507401	12.231286
1962	142.0	53.609249	12.097245
1967	142.0	55.678290	11.718858
1972	142.0	57.647386	11.381953
1977	142.0	59.570157	11.227229
1982	142.0	61.533197	10.770618
1987	142.0	63.212613	10.556285
1992	142.0	64.160338	11.227380
1997	142.0	65.014676	11.559439
2002	142.0	65.694923	12.279823
2007	142.0	67.007423	12.073021

In [386]:

```
# 使用字典在agg和aggregate中
gdf=df.groupby("year").agg({"lifeExp":np.mean,"pop":np.median,"gdpPercap":np.median}
gdf
```

Out[386]:

	lifeExp	pop	gdpPercap
year			
1952	49.057620	3943953.0	1968.528344
1957	51.507401	4282942.0	2173.220291
1962	53.609249	4686039.5	2335.439533
1967	55.678290	5170175.5	2678.334741
1972	57.647386	5877996.5	3339.129407
1977	59.570157	6404036.5	3798.609244
1982	61.533197	7007320.0	4216.228428
1987	63.212613	7774861.5	4280.300366
1992	64.160338	8688686.5	4386.085502
1997	65.014676	9735063.5	4781.825478
2002	65.694923	10372918.5	5319.804524
2007	67.007423	10517531.0	6124.371109

In [387]:

```
# 字典的agg应用至Series
gdf=df.groupby("year").lifeExp.agg([np.count_nonzero,np.mean,np.std])
gdf
```

Out[387]:

	count_nonzero	mean	std
year			
1952	142.0	49.057620	12.225956
1957	142.0	51.507401	12.231286
1962	142.0	53.609249	12.097245
1967	142.0	55.678290	11.718858
1972	142.0	57.647386	11.381953
1977	142.0	59.570157	11.227229
1982	142.0	61.533197	10.770618
1987	142.0	63.212613	10.556285
1992	142.0	64.160338	11.227380
1997	142.0	65.014676	11.559439
2002	142.0	65.694923	12.279823
2007	142.0	67.007423	12.073021

1.3 数据转换（标准化）

In [388]:

```
# z-score 计算
from scipy.stats import zscore
sp_z_grouped=df.groupby("year").lifeExp.transform(zscore)
sp_z_grouped
```

Out[388]:

```
0      -1.662719
1      -1.737377
2      -1.792867
3      -1.854699
4      -1.900878
          ...
1699   -0.081910
1700   -0.338167
1701   -1.580537
1702   -2.100756
1703   -1.955077
Name: lifeExp, Length: 1704, dtype: float64
```

1.4 过滤器filter

In [389]:

```
# 加载数据集
tips=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\seaborn-data-master
tips.shape
```

Out[389]:

```
(244, 7)
```

In [390]:

```
tips["size"].value_counts()
```

Out[390]:

```
2    156
3     38
4     37
5      5
6      4
1      4
Name: size, dtype: int64
```

In [391]:

```
# 过滤小于30的
tips_filter=tips.groupby("size").filter(lambda x: x["size"].count()>=30)
tips_filter
```

Out[391]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

231 rows × 7 columns

In [392]:

```
tips_filter.shape
```

Out[392]:

```
(231, 7)
```

In [393]:

```
# 成功过滤掉
tips_filter["size"].value_counts()
```

Out[393]:

```
2    156
3     38
4     37
Name: size, dtype: int64
```

2 第十一章：日期datetime数据类型

2.1 Python的datetime对象

In [394]:

```
# 显示现在的时间
from datetime import datetime
print(datetime.now())
```

```
2021-01-27 17:13:13.570958
```

In [395]:

```
# 求日期差值
t1=datetime.now()
t2=datetime(1995,12,24)
print(t1-t2)
t3=datetime(1968,6,1)
print(t1-t3)
```

```
9166 days, 17:13:13.587948
19233 days, 17:13:13.587948
```

2.2 转换datetime

In [396]:

```
import pandas as pd
ebola=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\data/country_times
ebola.head()
```

Out[396]:

	Date	Day	Cases_Guinea	Cases_Liberia	Cases_SierraLeone	Cases_Nigeria	Cases_Sen
0	1/5/2015	289	2776.0	NaN	10030.0	NaN	
1	1/4/2015	288	2775.0	NaN	9780.0	NaN	
2	1/3/2015	287	2769.0	8166.0	9722.0	NaN	
3	1/2/2015	286	NaN	8157.0	NaN	NaN	
4	12/31/2014	284	2730.0	8115.0	9633.0	NaN	

In [397]:

```
ebola.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 122 entries, 0 to 121
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Date                 122 non-null    object 
 1   Day                  122 non-null    int64  
 2   Cases_Guinea         93 non-null     float64
 3   Cases_Liberia        83 non-null     float64
 4   Cases_SierraLeone    87 non-null     float64
 5   Cases_Nigeria        38 non-null     float64
 6   Cases_Senegal        25 non-null     float64
 7   Cases_UnitedStates   18 non-null     float64
 8   Cases_Spain          16 non-null     float64
 9   Cases_Mali           12 non-null     float64
 10  Deaths_Guinea        92 non-null     float64
 11  Deaths_Liberia       81 non-null     float64
 12  Deaths_SierraLeone   87 non-null     float64
 13  Deaths_Nigeria       38 non-null     float64
 14  Deaths_Senegal       22 non-null     float64
 15  Deaths_UnitedStates  18 non-null     float64
 16  Deaths_Spain         16 non-null     float64
 17  Deaths_Mali          12 non-null     float64
dtypes: float64(16), int64(1), object(1)
memory usage: 17.3+ KB
```

In [398]:

```
# 使用to_datetime 进行转哈
ebola["date_dt"]=pd.to_datetime(ebola["Date"])
ebola.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 122 entries, 0 to 121
Data columns (total 19 columns):
 #   Column               Non-Null Count  Dtype         
---  ------               --------------  -----         
 0   Date                 122 non-null    object        
 1   Day                  122 non-null    int64         
 2   Cases_Guinea         93 non-null     float64       
 3   Cases_Liberia        83 non-null     float64       
 4   Cases_SierraLeone    87 non-null     float64       
 5   Cases_Nigeria        38 non-null     float64       
 6   Cases_Senegal        25 non-null     float64       
 7   Cases_UnitedStates   18 non-null     float64       
 8   Cases_Spain          16 non-null     float64       
 9   Cases_Mali           12 non-null     float64       
 10  Deaths_Guinea        92 non-null     float64       
 11  Deaths_Liberia       81 non-null     float64       
 12  Deaths_SierraLeone   87 non-null     float64       
 13  Deaths_Nigeria       38 non-null     float64       
 14  Deaths_Senegal       22 non-null     float64       
 15  Deaths_UnitedStates  18 non-null     float64       
 16  Deaths_Spain         16 non-null     float64       
 17  Deaths_Mali          12 non-null     float64       
 18  date_dt              122 non-null    datetime64[ns]
dtypes: datetime64[ns](1), float64(16), int64(1), object(1)
memory usage: 18.2+ KB
```

In [399]:

```
ebola["date_dt"]
```

Out[399]:

```
0     2015-01-05
1     2015-01-04
2     2015-01-03
3     2015-01-02
4     2014-12-31
         ...    
117   2014-03-27
118   2014-03-26
119   2014-03-25
120   2014-03-24
121   2014-03-22
Name: date_dt, Length: 122, dtype: datetime64[ns]
```

In [400]:

```
# 获取具体日期 即dt.year dt.month,dt.day
ebola["date_dt"].dt.year
```

Out[400]:

```
0      2015
1      2015
2      2015
3      2015
4      2014
       ...
117    2014
118    2014
119    2014
120    2014
121    2014
Name: date_dt, Length: 122, dtype: int64
```

In [401]:

```
ebola["date_dt"].dt.month
```

Out[401]:

```
0       1
1       1
2       1
3       1
4      12
       ..
117     3
118     3
119     3
120     3
121     3
Name: date_dt, Length: 122, dtype: int64
```

In [402]:

```
ebola["date_dt"].dt.day
```

Out[402]:

```
0       5
1       4
2       3
3       2
4      31
       ..
117    27
118    26
119    25
120    24
121    22
Name: date_dt, Length: 122, dtype: int64
```

2.3 日期运算和Timedelta

In [403]:

```
# 添加一列计算日期开始和截至的过程时间
ebola["outbreak_d"]=ebola["date_dt"]-ebola["date_dt"].mean()
ebola[["Date","date_dt","Day","outbreak_d"]]
```

Out[403]:

	Date	date_dt	Day	outbreak_d
0	1/5/2015	2015-01-05	289	144 days 03:44:15.737704960
1	1/4/2015	2015-01-04	288	143 days 03:44:15.737704960
2	1/3/2015	2015-01-03	287	142 days 03:44:15.737704960
3	1/2/2015	2015-01-02	286	141 days 03:44:15.737704960
4	12/31/2014	2014-12-31	284	139 days 03:44:15.737704960
...	...	...	...	...
117	3/27/2014	2014-03-27	5	-140 days +03:44:15.737704960
118	3/26/2014	2014-03-26	4	-141 days +03:44:15.737704960
119	3/25/2014	2014-03-25	3	-142 days +03:44:15.737704960
120	3/24/2014	2014-03-24	2	-143 days +03:44:15.737704960
121	3/22/2014	2014-03-22	0	-145 days +03:44:15.737704960

122 rows × 4 columns

In [404]:

```
# 使用datetime方法
banks=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\data/banklist.csv'
banks.head()
```

Out[404]:

	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date	Updated Date
0	Fayette County Bank	Saint Elmo	IL	1802	United Fidelity Bank, fsb	26-May-17	26-Jul-17
1	Guaranty Bank, (d/b/a BestBank in Georgia & Mi...	Milwaukee	WI	30003	First-Citizens Bank & Trust Company	5-May-17	26-Jul-17
2	First NBC Bank	New Orleans	LA	58302	Whitney Bank	28-Apr-17	26-Jul-17
3	Proficio Bank	Cottonwood Heights	UT	35495	Cache Valley Bank	3-Mar-17	18-May-17
4	Seaway Bank and Trust Company	Chicago	IL	19328	State Bank of Texas	27-Jan-17	18-May-17

In [405]:

```
banks=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\data/banklist.csv'
banks.head()
# 将文本数据读入成datetime
```

Out[405]:

	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date	Updated Date
0	Fayette County Bank	Saint Elmo	IL	1802	United Fidelity Bank, fsb	2017-05-26	2017-07-26
1	Guaranty Bank, (d/b/a BestBank in Georgia & Mi...	Milwaukee	WI	30003	First-Citizens Bank & Trust Company	2017-05-05	2017-07-26
2	First NBC Bank	New Orleans	LA	58302	Whitney Bank	2017-04-28	2017-07-26
3	Proficio Bank	Cottonwood Heights	UT	35495	Cache Valley Bank	2017-03-03	2017-05-18
4	Seaway Bank and Trust Company	Chicago	IL	19328	State Bank of Texas	2017-01-27	2017-05-18

In [406]:

```
# 添加2列，表示银行破产的年度和月份
banks["closing_quarter"],banks["closing_year"]=(banks["Closing Date"].dt.quarter,bar
banks
```

Out[406]:

	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date	Updated Date	closing_quarter	closing_
0	Fayette County Bank	Saint Elmo	IL	1802	United Fidelity Bank, fsb	2017-05-26	2017-07-26	2	2
1	Guaranty Bank, (d/b/a BestBank in Georgia & Mi...	Milwaukee	WI	30003	First-Citizens Bank & Trust Company	2017-05-05	2017-07-26	2	2
2	First NBC Bank	New Orleans	LA	58302	Whitney Bank	2017-04-28	2017-07-26	2	2
3	Proficio Bank	Cottonwood Heights	UT	35495	Cache Valley Bank	2017-03-03	2017-05-18	1	2
4	Seaway Bank and Trust Company	Chicago	IL	19328	State Bank of Texas	2017-01-27	2017-05-18	1	2
...	...	...	...	...	...	...	...	...	
548	Superior Bank, FSB	Hinsdale	IL	32646	Superior Federal, FSB	2001-07-27	2014-08-19	3	2
549	Malta National Bank	Malta	OH	6629	North Valley Bank	2001-05-03	2002-11-18	2	2
550	First Alliance Bank & Trust Co.	Manchester	NH	34264	Southern New Hampshire Bank & Trust	2001-02-02	2003-02-18	1	2
551	National State Bank of Metropolis	Metropolis	IL	3815	Banterra Bank of Marion	2000-12-14	2005-03-17	4	2
552	Bank of Honolulu	Honolulu	HI	21029	Bank of the Orient	2000-10-13	2005-03-17	4	2

553 rows × 9 columns

In [407]:

```
# 计算每年倒闭的银行数量，使用size函数
closing_year=banks.groupby(["closing_year"]).size()
closing_year
```

Out[407]:

```
closing_year
2000      2
2001      4
2002     11
2003      3
2004      4
2007      3
2008     25
2009    140
2010    157
2011     92
2012     51
2013     24
2014     18
2015      8
2016      5
2017      6
dtype: int64
```

In [408]:

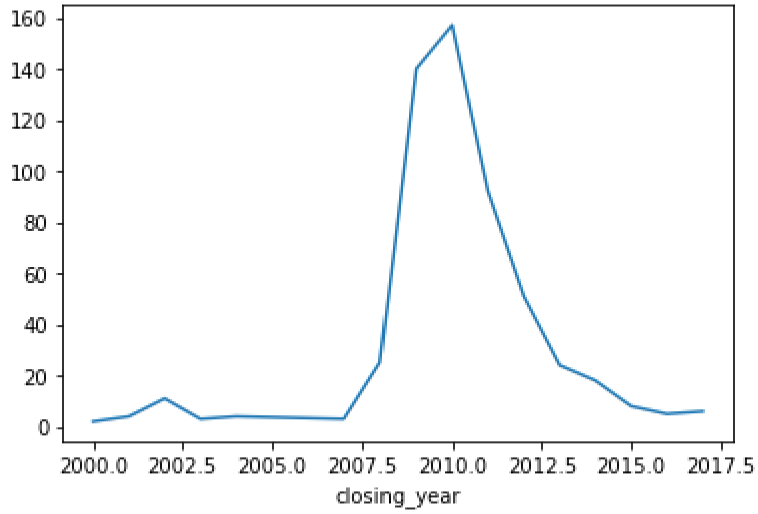
```
# 计算每个季度的银行倒闭数量，使用size函数
closing_quarter=banks.groupby(["closing_year","closing_quarter"]).size()
closing_quarter.head()
```

Out[408]:

```
closing_year  closing_quarter
2000          4                  2
2001          1                  1
              2                  1
              3                  2
2002          1                  6
dtype: int64
```

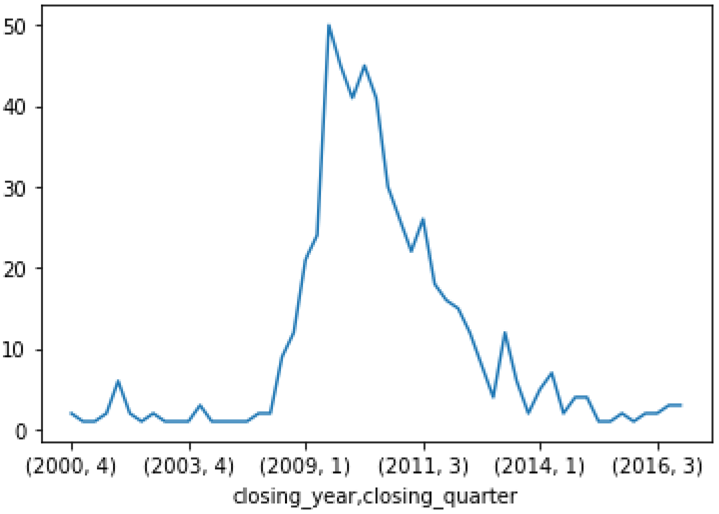
In [409]:

```
# 画图
import matplotlib.pyplot as plt
fig,ax=plt.subplots()
ax=closing_year.plot()
plt.show()
```



In [410]:

```
fig,ax=plt.subplots()
ax=closing_quarter.plot()
plt.show()
```



2.4 基于日期获取数据子集

In [411]:

```
# 加载股票数据集
tesla=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\data/tesla_stock_y
tesla.head()
```

Out[411]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-06-29	19.000000	25.00	17.540001	23.889999	23.889999	18766300
1	2010-06-30	25.790001	30.42	23.299999	23.830000	23.830000	17187100
2	2010-07-01	25.000000	25.92	20.270000	21.959999	21.959999	8218800
3	2010-07-02	23.000000	23.10	18.709999	19.200001	19.200001	5139800
4	2010-07-06	20.000000	20.00	15.830000	16.110001	16.110001	6866900

In [412]:

```
# 看到第一列为日期，可以重新读取，然后进行拆分日期
tesla=pd.read_csv("E:\jupyter notebook storage\Practice in Pandas\data/tesla_stock_y
tesla.head()
```

Out[412]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-06-29	19.000000	25.00	17.540001	23.889999	23.889999	18766300
1	2010-06-30	25.790001	30.42	23.299999	23.830000	23.830000	17187100
2	2010-07-01	25.000000	25.92	20.270000	21.959999	21.959999	8218800
3	2010-07-02	23.000000	23.10	18.709999	19.200001	19.200001	5139800
4	2010-07-06	20.000000	20.00	15.830000	16.110001	16.110001	6866900

In [413]:

```
# 获取日期子集,2010年6月份的数据
print(tesla.loc[(tesla["Date"].dt.year==2010)&(tesla["Date"].dt.month==6)])
```

```
         Date      Open   High        Low      Close  Adj Close     Vol
ume
0 2010-06-29  19.000000  25.00  17.540001  23.889999  23.889999  18766
300
1 2010-06-30  25.790001  30.42  23.299999  23.830000  23.830000  17187
100
```

In [414]:

```
# Timedeltaindex对象
tesla["ref_date"]=tesla["Date"]-tesla["Date"].min()
tesla.index=tesla["ref_date"]
tesla.head()
```

Out[414]:

	Date	Open	High	Low	Close	Adj Close	Volume	ref_date
ref_date								
0 days	2010-06-29	19.000000	25.00	17.540001	23.889999	23.889999	18766300	0 days
1 days	2010-06-30	25.790001	30.42	23.299999	23.830000	23.830000	17187100	1 days
2 days	2010-07-01	25.000000	25.92	20.270000	21.959999	21.959999	8218800	2 days
3 days	2010-07-02	23.000000	23.10	18.709999	19.200001	19.200001	5139800	3 days
7 days	2010-07-06	20.000000	20.00	15.830000	16.110001	16.110001	6866900	7 days

In [415]:

```
# 通过ref_date进行筛选,在index的状态下, 空格不影响选取结果
print(tesla["0 days":"5 days"])
```

```
               Date       Open   High        Low      Close  Adj Close  
\
ref_date                                                                  
0 days   2010-06-29  19.000000  25.00  17.540001  23.889999  23.889999   
1 days   2010-06-30  25.790001  30.42  23.299999  23.830000  23.830000   
2 days   2010-07-01  25.000000  25.92  20.270000  21.959999  21.959999   
3 days   2010-07-02  23.000000  23.10  18.709999  19.200001  19.200001   

            Volume ref_date  
ref_date                     
0 days    18766300   0 days  
1 days    17187100   1 days  
2 days     8218800   2 days  
3 days     5139800   3 days  
```

2.5 date_range函数使用

In [416]:

```
print(pd.date_range("2021-01-05","2021-1-15",freq="b"))
```

```
DatetimeIndex(['2021-01-05', '2021-01-06', '2021-01-07', '2021-01-08',
               '2021-01-11', '2021-01-12', '2021-01-13', '2021-01-14',
               '2021-01-15'],
              dtype='datetime64[ns]', freq='B')
```

freq参数的代码和所含意义：

B

business day frequency

C

custom business day frequency

D

calendar day frequency

W

weekly frequency

M

month end frequency

SM

semi-month end frequency (15th and end of month)

BM

business month end frequency

CBM

custom business month end frequency

MS

month start frequency

SMS

semi-month start frequency (1st and 15th)

BMS

business month start frequency

CBMS

custom business month start frequency

Q

quarter end frequency

BQ

business quarter end frequency

QS

quarter start frequency

BQS

business quarter start frequency

A, Y

year end frequency

BA, BY

business year end frequency

AS, YS

year start frequency

BAS, BYS

business year start frequency

BH

business hour frequency

H

hourly frequency

T, min

minutely frequency

S

secondly frequency

L, ms

milliseconds

U, us

microseconds

N

nanoseconds

In [417]:

```
# 偏移量，在代码前加数字即可，表示间隔
print(pd.date_range("2021-01-05","2021-1-15",freq="2b"))
```

```
DatetimeIndex(['2021-01-05', '2021-01-07', '2021-01-11', '2021-01-13',
               '2021-01-15'],
              dtype='datetime64[ns]', freq='2B')
```

2.6 时区

In [418]:

```
# 时区对于中国来说用处不大，即中国统一北京时间，对于国外，可以利用pytz包来实现
import pytz
print(len(pytz.all_timezones))
```

```
593
```

In [419]:

```
# 查找美国时区
import re
regex=re.compile(r'^US')
selected_files=filter(regex.search,pytz.common_timezones)
print(list(selected_files))
```

```
['US/Alaska', 'US/Arizona', 'US/Central', 'US/Eastern', 'US/Hawaii',
'US/Mountain', 'US/Pacific']
```

In [420]:

```
# 假设为美国太平洋时间
arrive=pd.Timestamp("2021-1-1 10:22")
arrive=arrive.tz_localize('US/Pacific')
# 换时间为美东
print(arrive.tz_convert('US/Eastern'))
# 特别的，计算差值时，需转换相同时区，或者无时区，才可进行计算
```

```
2021-01-01 13:22:00-05:00
```